Univerza *v Ljubljani*

Skripta za vaje predmeta Bayesova statistika (z rešitvami)

za interdisciplinarni magistrski študij Uporabna statistika in druge magistrske študije Fakultete za matematiko in fiziko Univerze v Ljubljani

Nina Ružić Gorenjec

Ljubljana, 2026

Kazalo

Predgovor

V skripti je združeno celotno gradivo za (aplikativne) vaje predmeta Bayesova statistika, ki se izvaja na naslednjih magistrskih študijih Univerze v Ljubljani:

- na interdisciplinarnem magistrskem študiju Uporabna statistika (kot obvezni predmet za modula Matematična statistika in Strojno učenje, ter kot izbirni predmet za preostale module) je to gradivo za celotne vaje;
- na magistrskih študijih Fakultete za matematiko in fiziko Oddelka za matematiko (študiji Matematika, Finančna matematika, Interdisciplinarni študij računalništvo in matematika) se predmet izvaja kot izbirni predmet in ima poleg aplikativnih vaj tudi teoretične vaje, ki niso zajete v teh zbirki vaj.

Gradivo je nastalo v študijskem letu 2019/2020 in se izpopolnjevalo vse do študijskega leta 2023/2024. Gradivo je bilo uporabljeno tudi v letošnjem študijskem letu 2025/2026.

Skripta ob vajah vsebuje tudi teoretične povzetke s predavanj, ki so potrebni za razumevanje, oznake v skripti so usklajene s predavanji in (večinoma) s knjigo P. D. Hoff, *A First Course in Bayesian Statistical Methods* (2009). Skripta vsebuje vse rešitve, vključno z izvorno R kodo in vsemi izpisi.

1. sklop: Binomski model

1 Primer

Izberite pravilni odgovor na spodnje vprasanje.

Vprasanje: Qskd senciljm dowdlq a?

(a) 25

(b) 625

(c) 1

(d) Nic od nastetega.

Zanima nas verjetnost, da odgovorimo pravilno.

2 Verjetnostni model za nas primer

Vzorec $X_1, X_2, \ldots, X_n$, kjer je:

- n stevilo studentov na vajah,
- X_i predstavlja pravilnost odgovora i-tega studenta, tj. $X_i = 1$, ce i-ti student odgovori pravilno, in $X_i = 0$, ce le-ta odgovori napacno.

Preucujemo $X_1 + X_2 + \ldots + X_n$, tj. stevilo vseh pravilnih odgovorov, ki ga oznacimo z X (druga standardna oznaka je Y v smislu izida, anglesko *outcome*).

- $X \mid \theta \sim \text{Bin}(n, \theta)$
- $P(X = k \mid \theta) = \binom{n}{k}\theta^k(1-\theta)^{n-k}$
- θ (na predavanjih ϑ) je verjetnost pravilnega odgovora – **parameter, ki nas zanima**
- $\text{E}(X) = n\theta$

Nas primer:

```
n <- 26
```

Nasi podatki (oznacimo s k realizacijo X na nasem vzorcu):

```
k <- 6
```

2.1 Kako bi ocenili nas parameter s "klasicno" frekventisticno statistiko? Katere metode bi lahko uporabili?

Uporabili bi metodo največjega verjetja ali metodo momentov, z obema metodama bi dobili k/n.

2.2 Bayesova formula

Na ravni dogodkov:

$$P(A \mid B) = \frac{P(B \mid A)\, P(A)}{P(B)} \propto P(B \mid A)\, P(A).$$

Na ravni gostot z enotno oznako p (lahko bi pisali tudi f):

$$p(\theta \mid \text{podatki}) = \frac{p(\text{podatki} \mid \theta)\, p(\theta)}{p(\text{podatki})} \propto p(\text{podatki} \mid \theta)\, p(\theta).$$

V "standardnih" oznakah:

$$\pi(\theta \mid x) = \frac{f(x \mid \theta)\, \pi(\theta)}{f(x)} \propto f(x \mid \theta)\, \pi(\theta).$$

Podatki x so znani. Zanima nas θ. **Poglejmo zato na zgornje kot na funkcijo θ.**

Spomnimo se funkcije verjetja (anglesko *likelihood*) $L(\theta \mid x) = L(\theta\,; x)$ in zapisimo zgornje kot:

$$\pi(\theta \mid x) \propto L(\theta \mid x)\, \pi(\theta).$$

Trije gradniki Bayesove formule, ki jih bomo predstavili graficno kot funkcije θ:

- **Apriorna porazdelitev** $\pi(\theta)$ (njen integral je 1) – *nase vnaprejsnje (apriorno) vedenje/prepricanje o θ, preden zberemo podatke!*
- **Verjetje** $L(\theta \mid x)$ (potrebno mnoziti s konstanto, tako da bo integral enak 1) – *verjetnost podatkov pri razlicnih moznih vrednostih parametra θ.*
- **Aposteriorna porazdelitev** $\pi(\theta \mid x)$ – *preko Bayesove formule posodobimo nase apriorno vedenje o parametru ($\pi(\theta)$) s tem, kar nam povedo podatki ($L(\theta \mid x)$).*

2.3 Verjetje

Stevilo pravilnih odgovorov med n studenti je enako k:

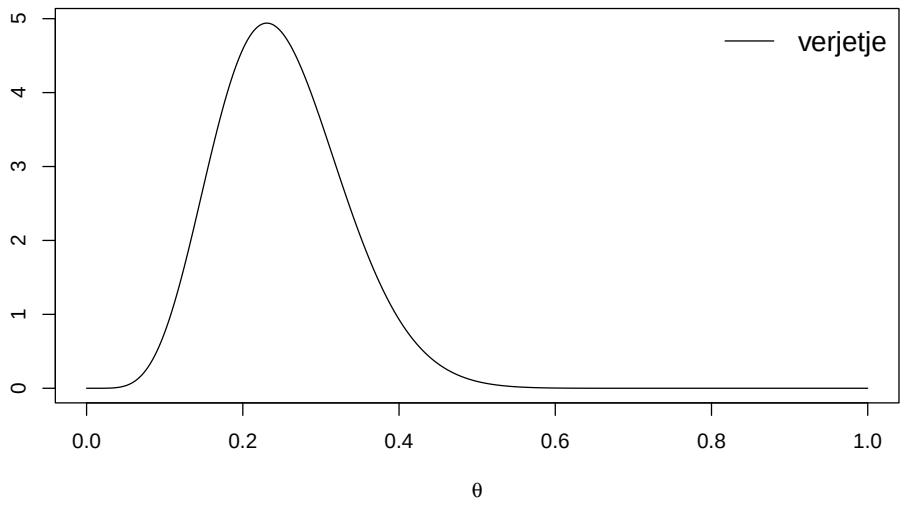
$$L(\theta \mid x) = \theta^k \, (1-\theta)^{n-k}.$$

V R-u:

```
verjetje <- function(theta, k, n){
  dbinom(k, size = n, prob = theta)
}

#Z mnozenjem s konst dosezemo, da je integral verjetja glede na theta enak 1.
konst <- function(k, n){
  theta <- seq(0.001, 1, 0.001)
  1 / (0.001 * sum(verjetje(theta, k, n)))
}
```

Narisemo za nas vzorec:

```
theta <- seq(0, 1, 0.001)
konst.verjetje <- konst(k, n) * verjetje(theta, k, n)
plot(theta, konst.verjetje, type = "l",
     xlab = expression(theta), ylab ="")
legend("topright", legend = c("verjetje"), col = c("black"),
       lty = 1, bty = "n", cex = 1.3)
```



2.4 Apriorna porazdelitev

Za apriorno porazdelitev si izberemo beta porazdelitev, ki je v primeru binomske porazdelitve podatkov *conjugate prior* (pomeni, da apriorna in aposteriorna porazdelitev pripadata enaki druzini porazdelitev), zato se lahko uporablja tudi izraz **beta-binomski model**.

Za apriorno porazdelitev imamo torej gostoto beta porazdelitve pri parametrih $\alpha, \beta > 0$:

$$\pi(\theta) = \pi(\theta \mid \alpha, \beta) = \frac{1}{B(\alpha, \beta)} \theta^{\alpha-1}(1-\theta)^{\beta-1},$$

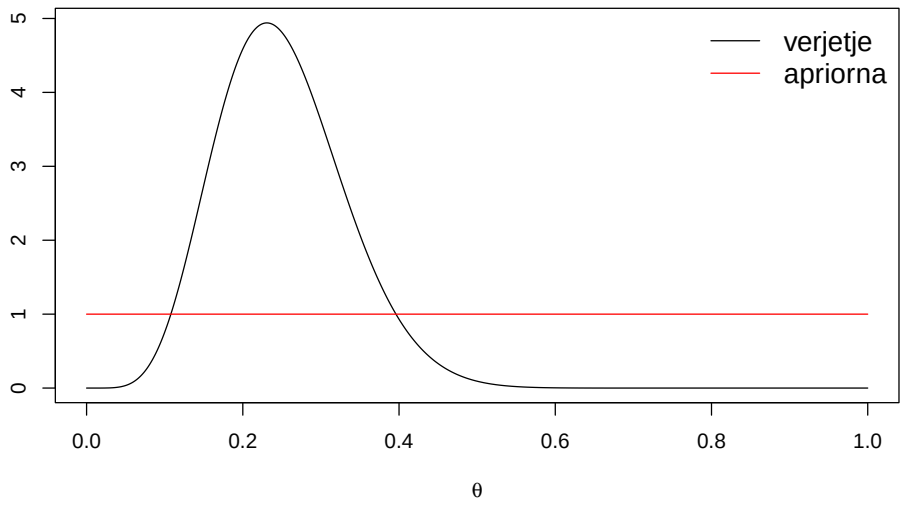
kjer je funkcija beta $B(\alpha, \beta) = \frac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha+\beta)}$ in je funkcija gama $\Gamma(a) = (a-1)!$ za pozitivna cela stevila a. Spomnimo se:

- $\mathrm{E}(\mathrm{Beta}(\alpha, \beta)) = \frac{\alpha}{\alpha+\beta}$,
- $\mathrm{var}(\mathrm{Beta}(\alpha, \beta)) = \frac{\alpha\beta}{(\alpha+\beta)^2(\alpha+\beta+1)}$.

Najprej poskusimo $\alpha = \beta = 1$, s cimer dobimo enakomerno zvezno porazdelitev U[0,1] - **neinformativna apriorna porazdelitev**.

Narisemo v R-u:

```
alpha <- 1
beta <- 1

theta <- seq(0, 1, 0.001)
apriorna <- dbeta(theta, alpha, beta)

konst.verjetje <- konst(k, n) * verjetje(theta, k, n)

y.max <- max(c(konst.verjetje, apriorna))
plot(theta, konst.verjetje, ylim = c(0, y.max), type = "l",
     xlab = expression(theta), ylab = "")
lines(theta, apriorna, col = "red")
legend("topright", legend = c("verjetje", "apriorna"), col = c("black", "red"),
       lty = 1, bty = "n", cex = 1.3)
```



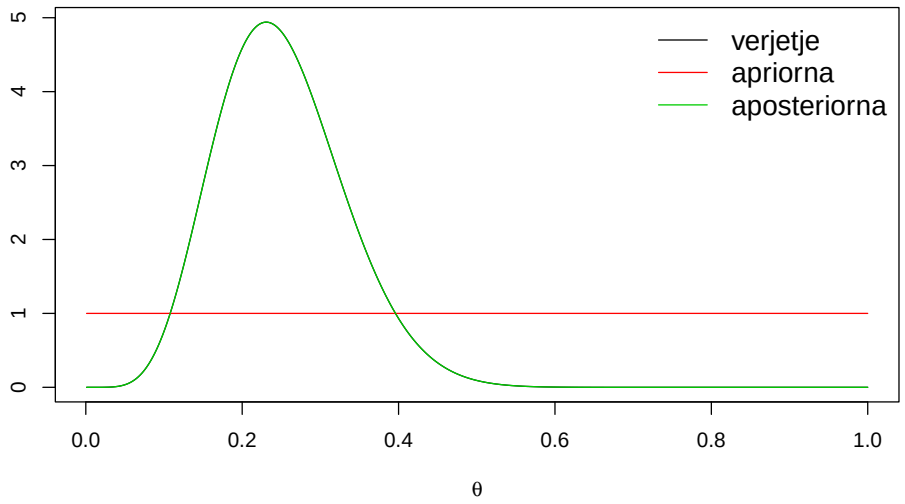
2.5 Aposteriorna porazdelitev

Ker smo uporabili *conjugate prior*, bo aposteriorna porazdelitev tudi iz druzine beta porazdelitev, njena parametra sta enaka:

- $\alpha_{\text{apost}} = k + \alpha$,
- $\beta_{\text{apost}} = n - k + \beta$.

Narisemo v R-u:

```
alpha.apost <- k + alpha
beta.apost <- n - k + beta

theta <- seq(0.001, 1, 0.001)
aposteriorna <- dbeta(theta, alpha.apost, beta.apost)

konst.verjetje <- konst(k, n) * verjetje(theta, k, n)
apriorna <- dbeta(theta, alpha, beta)

y.max <- max(c(konst.verjetje, apriorna, aposteriorna))
plot(theta, konst.verjetje, ylim=c(0, y.max), type = "l",
     xlab = expression(theta), ylab = "")
lines(theta, apriorna, col = "red")
lines(theta, aposteriorna, col = "green3")
legend("topright", legend = c("verjetje", "apriorna","aposteriorna"),
       col = c("black","red","green3"), lty = 1, bty = "n", cex = 1.3)
```



2.6 Ocena parametra θ

Ena moznost je pricakovana vrednost aposteriorne porazdelitve:

$$\hat{\theta} = \frac{\alpha_{\text{apost}}}{\alpha_{\text{apost}} + \beta_{\text{apost}}} = \frac{k + \alpha}{(k + \alpha) + (n - k + \beta)} = \frac{k + \alpha}{n + \alpha + \beta}.$$

```
alpha.apost / (alpha.apost + beta.apost)
```

```
## [1] 0.25
```

Ali dobimo enako kakor pri frekventisticnemu pristopu?

```
k/n
```

```
## [1] 0.2307692
```

V primeru neinformativne porazdelitve $\alpha = \beta = 1$, dobimo

$$\hat{\theta} = \frac{k + 1}{n + 2}.$$

Na predavanjih ste $\hat{\theta}$, ocenjen preko pricakovane vrednosti aposteriorne porazdelitve, zapisali kot

$$\hat{\theta} = \frac{\phi}{\phi + n} \cdot \mu + \frac{n}{\phi + n} \cdot \frac{k}{n},$$

kjer je $\mu = \text{E}(\text{Beta}(\alpha, \beta)) = \frac{\alpha}{\alpha + \beta}$ in $\phi = \alpha + \beta$.

Ideja: $\hat{\theta}$ je utezeno povprecje med $\text{E}(\text{apriorna})$ in $\text{E}(X)$, kjer preko ϕ kontroliramo, kako mocno verjamemo apriorni pricakovani vrednosti.

2.7 Interval (območje) zaupanja v Bayesovi statistiki

Pri danih podatkih $X = x$ ima interval $[L_B(x), U_B(x)]$ 95% **Bayesovo pokritje** za θ, če velja

$$P(L_B(x) < \theta < U_B(x) \mid X = x) = 0,95.$$

Preden zberemo podatke ima interval $[L(X), U(X)]$ 95% **frekventisticno pokritje** za θ, če velja

$$P(L(X) < \theta < U(X) \mid \theta) = 0,95.$$

Ko poznamo podatke $X = x$, jih vstavimo v $L(X)$ in $U(X)$ ter s tem dobimo $L(x)$ in $U(x)$.

Koliko je $P(L(x) < \theta < U(x) \mid \theta)$?

0 ali 1 :)

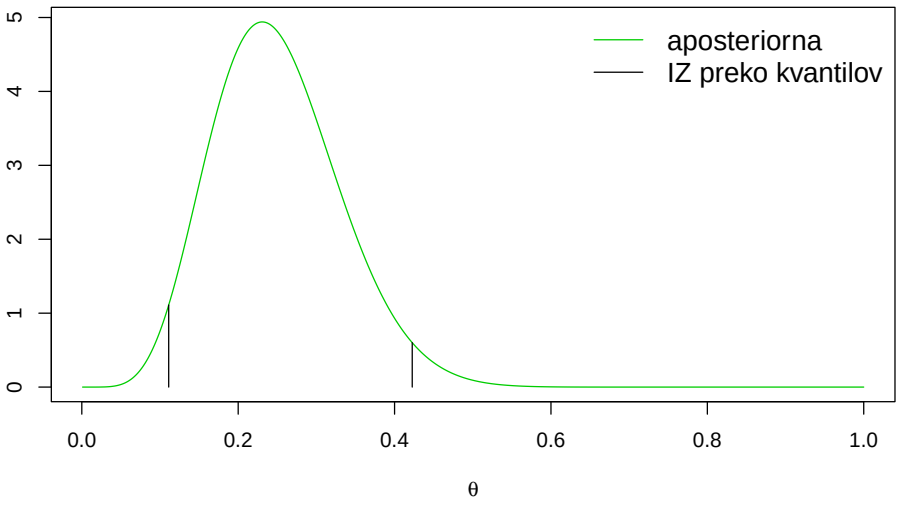
Interval zaupanja v Bayesovi statistiki (angl. pogosto *credible interval*, lahko tudi *confidence interval*) je torej katerikoli interval, v katerem "je vsebovanih 95% gostote aposteriorne porazdelitve". Seveda pa želimo, da je "centralen glede na porazdelitev".

Najbolj preprosta varianta preko kvantilov porazdelitve (smiselna, če je porazdelitev približno simetricna):

```
(iz <- qbeta(c(0.025,0.975),alpha.apost,beta.apost))
```

```
## [1] 0.1111446 0.4225831
```

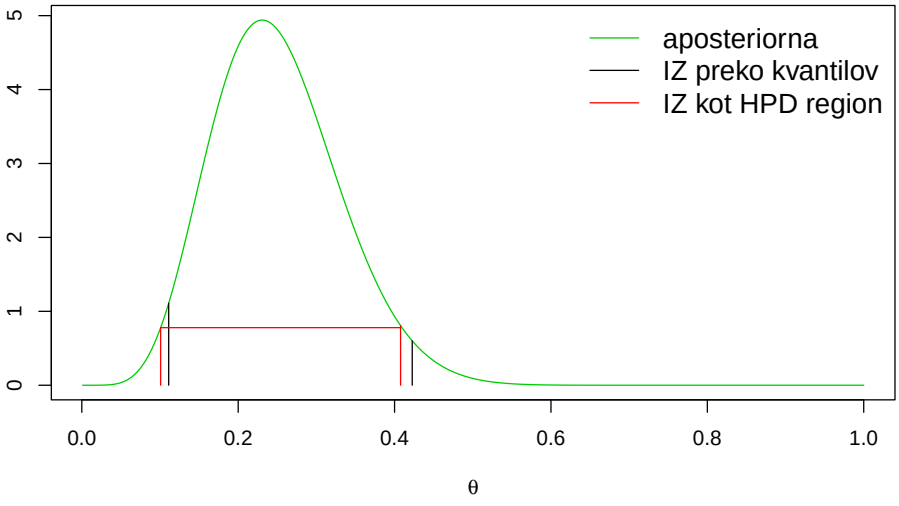
```
plot(theta, aposteriorna, type = "l", col="green3",
     xlab = expression(theta), ylab = "")
segments(x0 = iz[1], y0 = 0,
         x1 = iz[1], y1 = dbeta(iz[1], alpha.apost, beta.apost))
segments(x0 = iz[2], y0 = 0,
         x1 = iz[2], y1 = dbeta(iz[2], alpha.apost, beta.apost))
legend("topright", legend = c("aposteriorna","IZ preko kvantilov"),
       col = c("green3","black"), lty = 1, bty = "n", cex = 1.3)
```



Highest posterior density (HPD) region:

```
#install.packages("HDInterval")
library(HDInterval)

aposteriorna.sample <- rbeta(100000, alpha.apost, beta.apost)
(iz.hdi <- hdi(aposteriorna.sample, credMass = 0.95))
```

```
##     lower     upper
## 0.1008035 0.4076840
## attr(,"credMass")
## [1] 0.95
```



Kaksen interval zaupanja dobimo s frekventisticnim pristopom? Katere intervale zaupanja imamo na voljo?

```
prop.test(k, n, correct=F)$conf #IZ z aproksimacijo normalne porazdelitve
```

```
## [1] 0.1103385 0.4205155
## attr(,"conf.level")
## [1] 0.95
```

```
binom.test(k, n)$conf #Clopper-Pearsonov IZ
```

```
## [1] 0.08974011 0.43647510
## attr(,"conf.level")
## [1] 0.95
```

```
iz #Bayesov IZ, metoda s kvantili
```

```
## [1] 0.1111446 0.4225831
```

2.8 Testiranje hipotez

Namen stirih moznih odgovorov je bil, da je verjetnost pravilnega odgovora brez kakrsnegakoli ucenja dovolj majhna. Ali lahko sklepamo, da je verjetnost pravilnega odgovora manjsa od 0,4?

Kako testiramo to hipotezo s Bayesovim pristopom? Kaj mora biti nicelna in kaj alternativna hipoteza?

Kako testiramo s frekventisticnim pristopom?

Pri **Bayesovem pristopu** ni ničelne ali alternativne hipoteze, "preprosto" izračunamo verjetnost hipoteze glede na izračunano aposteriorno porazdelitev.

Pri **frekventističnem pristopu** postavimo ničelno domnevo in izračunamo vrednost p, tj. verjetnost, da dobimo izračunano testno statistiko na podlagi vzorca ali še bolj ekstremno, če velja ničelna domneva. Će je vrednost p premajhna, tipično pod 0,05, "ne verjamemo, da ničelna domneva drži" in jo zato zavrnemo ter potrdimo alternativno domnevo z verjetnostjo napačnega sklepa pod 5 %.

```r
#Bayesovski pristop
pbeta(0.4, alpha.apost, beta.apost)
```

```
## [1] 0.9579073
```

```r
#Test z aproksimacijo normalne porazdelitve
prop.test(k, n, p = 0.4, alternative = "less", correct = FALSE)$p.value
```

```
## [1] 0.03908454
```

```r
#Binomski eksaktni test
binom.test(k, n, p = 0.4, alternative = "less")$p.value
```

```
## [1] 0.05588404
```

3 Primerjava dveh neodvisnih delezev

Nase vprasanje iz zacetka navodil tega sklopa zastavimo se skupini studentov, ki se je ucila.

Od 30 studentov, jih je 21 odgovorilo pravilno.

Verjetnost pravilnega odgovora za studenta, ki se uci, naj bo θ_{uci}. Ocenimo jo z uporabo neinformativne apriorne porazdelitve (tako kakor prej, $\alpha = \beta = 1$). Dobimo aposteriorno porazdelitev

$$\theta_{\text{uci}} \sim \text{Beta}(21 + 1, 30 - 21 + 1) = \text{Beta}(22, 10).$$

Ocenimo $\hat{\theta}_{\text{uci}} = 22/(22 + 10) = 0,6875$.

Kaj smo predpostavili, da smo lahko uporabili ta model? Neodvisnost in enako porazdeljenost.

Prej smo ocenili verjetnost pravilnega odgovora za studenta, ki nakljucno izbere pravilni odgovor, kot

$$\theta_{\text{blef}} \sim \text{Beta}(6 + 1, 26 - 6 + 1) = \text{Beta}(7, 21).$$

Ocenimo $\hat{\theta}_{\text{blef}} = 7/(7 + 21) = 0,25$.

Zelimo primerjati ti dve verjetnosti.

Na kaksne nacine ju lahko primerjamo? Katere mere povezanosti lahko izracunamo?

Glejte naslednjo stran.

- Razlika delezev (angl. *risk difference*): $\theta_{\text{uci}} - \theta_{\text{blef}}$.
- Relativno tveganje (angl. *risk ratio*): $\theta_{\text{uci}}/\theta_{\text{blef}}$.
- Razmerje obetov (angl. *odds ratio*):

$$\frac{\theta_{\text{uci}}/(1-\theta_{\text{uci}})}{\theta_{\text{blef}}/(1-\theta_{\text{blef}})}.$$

Kako ocenimo vsako izmed teh mer?

Ocenimo parametre (dodamo strehe v notaciji).

Ali znate s orodji frekventisticne statistike testirati, da je razlika delezev vecja od nic?

Da, uporabimo test za primerjavo dveh neodvisnih deležev (obstaja več različic).

Kako bi testirali, da je relativno tveganje vecje od ena (studentje, ki se ucijo, imajo torej vecjo verjetnost pravilnega odgovora)? Kako bi izracunali interval zaupanja za relativno tveganje?

Frekventisticno?

Bayesovsko?

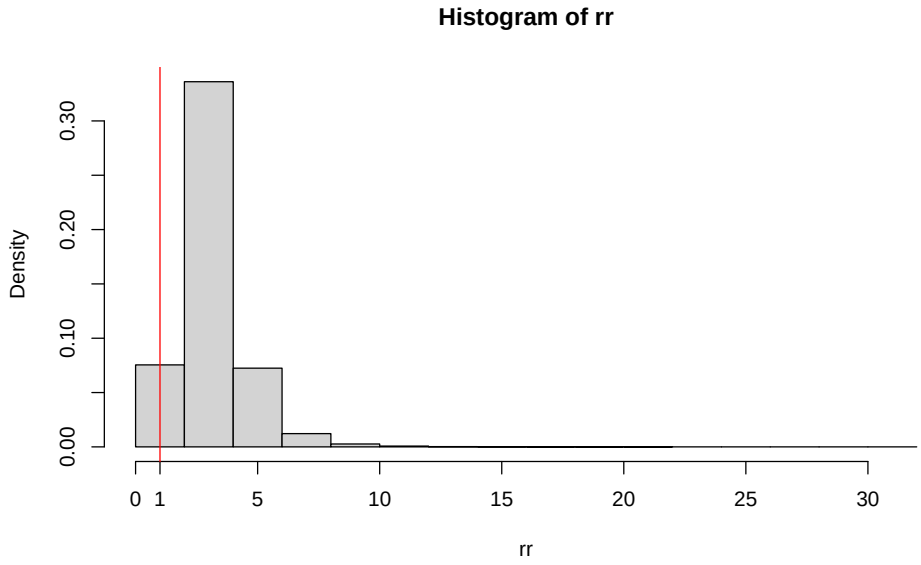
Moramo pri teh dveh pristopih kaj izpeljati?

Frekventistično: predlagati bi morali smiselno testno statistiko in izpeljati njeno porazdelitev pod ničelno domnevo.

Bayesovsko: lahko simuliramo iz dobljenih aposteriornih porazdelitev za posamezen parameter in izračunamo relativno tveganje, s čimer dobimo porazdelitev, nato oceno, interval in verjetnosti hipotez (ni dodatnih izpeljav).

```r
uci <- rbeta(1000000, 22, 10)
blef <- rbeta(1000000, 7, 21)

rr <- uci/blef
```

```r
hist(rr, prob = TRUE)
axis(1, at = 1)
abline(v = 1, col = "red")
```




```r
## Ocena:
mean(rr)
```

```
## [1] 3.092981
```

```r
## Interval zaupanja:
quantile(rr, probs = c(0.025, 0.975)) # preko kvantilov
```

```
##     2.5%    97.5%
## 1.522764 6.326747
```

```r
hdi(rr, credMass = 0.95) #preko HPD region
```

```
##    lower    upper
## 1.275754 5.570381
## attr(,"credMass")
## [1] 0.95
```

```r
## Verjetnost hipoteze, da ucenje pomaga:
sum(rr>1)/length(rr)
```

```
## [1] 0.999765
```

4 Napovedovanje (angl. *prediction*)

Izpit je sestavljen iz desetih vprasanj (taksnih iz zacetka navodil tega sklopa).

1. Denimo, da bi pred zacetkom prvih vaj dali izpit v resevanje nekemu studentu. Kaj lahko povemo o porazdelitvi stevila njegovih pravilnih odgovorov?
2. Na prvih vajah smo pridobili vzorec, s katerim smo preizkusili, kako na vprasanje odgovarjamo, ce ne znamo cisto nic. Vzorec ste bili studentje, prisotni na prvih vajah. Izpit damo v resevanje studentu, **ki ni bil prisoten na prvih vajah** in se tudi ni ucil. Kaj lahko povemo o porazdelitvi stevila njegovih pravilnih odgovorov?

Odgovor na 1. vprasanje je **apriorna napovedna porazdelitev** (angl. *prior predictive distribution*).
Ta nas tipicno ne zanima.

Odgovor na 2. vprasanje je **aposteriorna napovedna porazdelitev** (angl. *posterior predictive distribution*).

Splosna formula za apriorno napovedno porazdelitev:

$$f(x_{\text{nov}}) = \int_\Theta f(x_{\text{nov}}, \theta)\, d\theta = \int_\Theta f(x_{\text{nov}} \mid \theta)\pi(\theta)\, d\theta.$$

Splosna formula za aposteriorno napovedno porazdelitev:

$$f(x_{\text{nov}} \mid x) = \int_\Theta f(x_{\text{nov}}, \theta \mid x)\, d\theta = \int_\Theta f(x_{\text{nov}} \mid \theta, x)\pi(\theta \mid x)\, d\theta = \int_\Theta f(x_{\text{nov}} \mid \theta)\pi(\theta \mid x)\, d\theta.$$

V nasem modelu (binomski model z apriorno beta porazdelitvijo) je:

- $\pi(\theta) \sim \text{Beta}(\alpha, \beta)$; izbrali smo $\alpha = 1, \beta = 1$
- $\pi(\theta \mid x) \sim \text{Beta}(\alpha_{\text{apost}}, \beta_{\text{apost}}) = \text{Beta}(k + \alpha, n - k + \beta)$; za nas vzorec velikosti $n = 26$ smo dobili $k = 6$
- za $x_{\text{nov}} \equiv K \in \{0, 1, \ldots, N\}$ je $f(x_{\text{nov}} \mid \theta) = \binom{N}{K}\theta^K(1-\theta)^{N-K}$; dolocili smo $N = 10$, zanimajo nas vsi mozni K

Izkaze se, da je iskana apriorna ali aposteriorna napovedna porazdelitev iz druzine t.i. **beta-binomske porazdelitve** (BetaBin). To je diskretna porazdelitev Y s parametri $N \in \mathbb{N}$ in $\tilde{\alpha}, \tilde{\beta} > 0$, ki lahko zavzame vrednosti $K \in \{0, 1, \ldots, N\}$ in je

$$P(Y = K) = \binom{N}{K} \frac{B(K + \tilde{\alpha}, N - K + \tilde{\beta})}{B(\tilde{\alpha}, \tilde{\beta})}.$$

Apriorna napovedna porazdelitev v binomskem modelu: $\text{BetaBin}(N, \alpha, \beta)$.

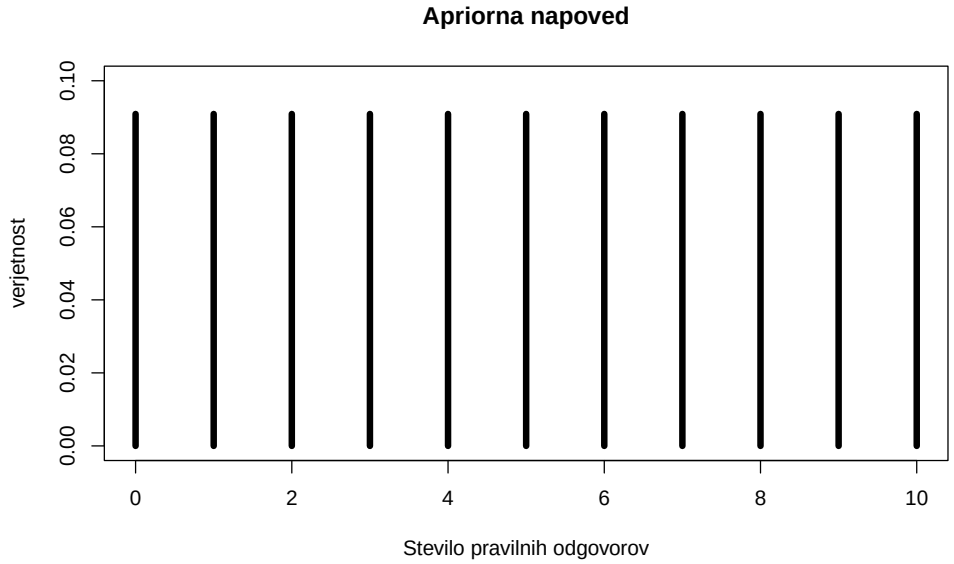
Aposteriorna napovedna porazdelitev v binomskem modelu: $\text{BetaBin}(N, \alpha_{\text{apost}}, \beta_{\text{apost}})$ oziroma $\text{BetaBin}(N, k + \alpha, n - k + \beta)$.

Beta-binomska porazdelitev v R (je vkljucena tudi v nekaterih paketih, ponekod drugace parametrizirana):

```
dbetabinom <- function(K, N, a, b){
  choose(N, K) * beta(K+a, N-K+b) / beta(a, b)
}
```

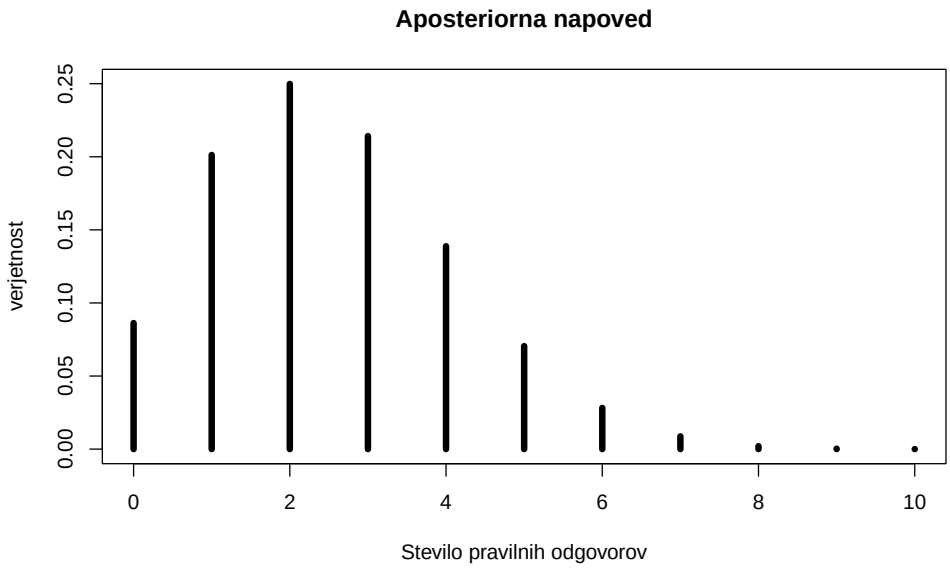
Narisemo apriorno napovedno porazdelitev.

```
plot(0:10, dbetabinom(0:10, N = 10, a = alpha, b = beta), type = "h",
     xlab = "Stevilo pravilnih odgovorov", ylab = "verjetnost",
     main = "Apriorna napoved", ylim = c(0, 0.1), lwd = 5)
```



Narisemo aposteriorno napovedno porazdelitev.

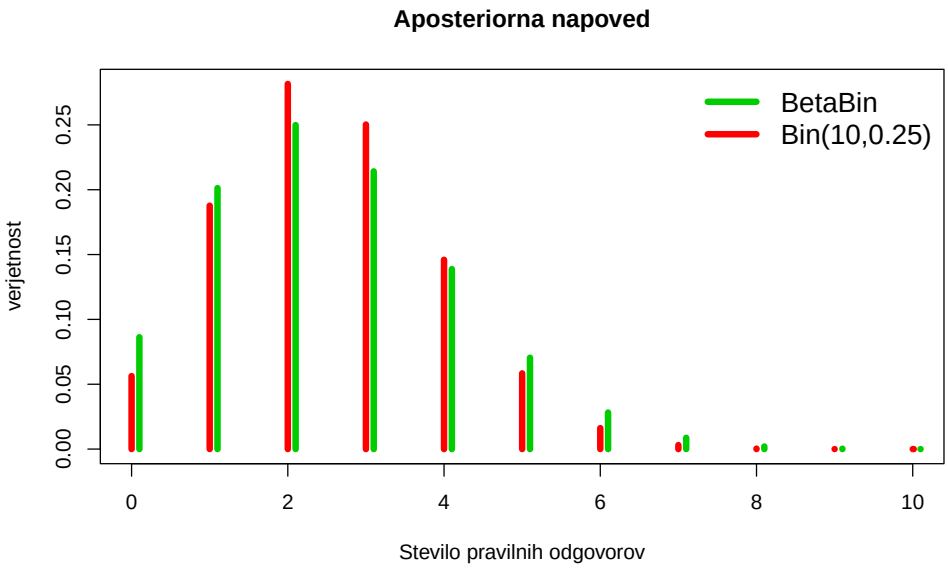
```
plot(0:10, dbetabinom(0:10, N = 10, a = alpha.apost, b = beta.apost), type = "h",
     xlab = "Stevilo pravilnih odgovorov", ylab = "verjetnost",
     main = "Aposteriorna napoved", lwd = 5)
```



Ali je bilo vse to racunanje res potrebno?

- Nasa ocena parametra po upostevanju podatkov nasega vzorca je $\hat{\theta} = \alpha_{\text{apost}}/(\alpha_{\text{apost}} + \beta_{\text{apost}}) = 0.25$.
- Stevilo pravilnih odgovorov je porazdeljeno $\text{Bin}(10, \theta)$.
- Ali je preprosto aposteriorna porazdelitev kar $\text{Bin}(10, \hat{\theta})$?

```
plot(0:10, dbinom(0:10, 10, alpha.apost / (alpha.apost + beta.apost)), type = "h",
     xlab = "Stevilo pravilnih odgovorov", ylab = "verjetnost",
     main = "Aposteriorna napoved", col = "red", lwd = 5)
segments(x0 = seq(0.1,10.1,1), y0 = rep(0,11),
         x1 = seq(0.1,10.1,1), y1 = dbetabinom(0:10, N = 10, a = alpha.apost, b = beta.a
         lwd = 5, col = "green3")
legend("topright",  lty = 1, lwd = 5,
       c("BetaBin", paste("Bin(10,",round(alpha.apost / (alpha.apost + beta.apost),2),")
       col = c("green3","red"), bty = "n", cex = 1.3)
```

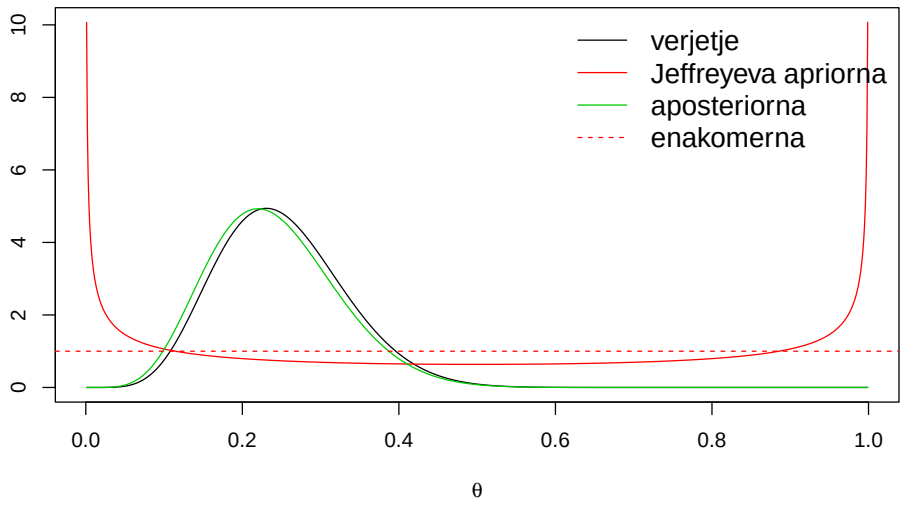

Aposteriorna napoved
BetaBin
Bin(10,0.25)
verjetnost
0.00
0.05
0.10
0.15
0.20
0.25
0
2
4
6
8
10
Stevilo pravilnih odgovorov

5 Jeffreyeva apriorna porazdelitev

To je **neinformativna** apriorna porazdelitev, ki je proporcionalna $\sqrt{\det \mathcal{I}(\theta)}$. Zanjo je znacilno, da je invariantna glede na razlicne reparametrizacije prostora parametrov.

Pri binomskem modelu je Jeffreyeva apriorna porazdelitev Beta($\alpha = 0.5$, $\beta = 0.5$).

Pri nasih podatkih dobimo:



2. sklop: Poissonov model

1 Primer

Zgodovinski podatki o stevilu rojstev cetverckov na leto v Prusiji za obdobje 69 let (Ladislaus von Bortkiewicz), za katere je znano, da se dobro prilegajo Poissonovi porazdelitvi.

```
(podatki <- data.frame(stevilo.cetverckov = 0:6,
                       stevilo.let = c(14,24,17,9,2,2,1)))
```

```
##   stevilo.cetverckov stevilo.let
## 1                  0          14
## 2                  1          24
## 3                  2          17
## 4                  3           9
## 5                  4           2
## 6                  5           2
## 7                  6           1
```

Zanima nas povprecno stevilo rojstev cetverckov na leto.

2 Verjetnostni model za nas primer

Vzorec $X_1, X_2, \ldots, X_n$, kjer je:

- $n = 69$ stevilo let,
- X_i predstavlja stevilo rojstev cetverckov v i-tem letu,
- $X_i \mid \theta \sim \text{Poiss}(\theta)$,
- $P(X_i = k \mid \theta) = \frac{1}{k!}\theta^k e^{-\theta}$ za $k \in \{0, 1, 2, \ldots\}$,
- $\text{E}(X_i) = \theta$ – **parameter, ki nas zanima**,
- $\text{var}(X_i) = \theta$.

Kaj so v nasem primeru X_i oz. njihova realizacija?

```
(x <- rep(podatki$stevilo.cetverckov, podatki$stevilo.let))
```

```
##  [1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [39] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 4 4 5 5 6
```

Vse in se vec o Poissonovi porazdelitvi je napisal doc. dr. Gaj Vidmar:

- clanek: http://ims.mf.uni-lj.si/archive/17(2)/31.pdf
- pripadajoce izracune lahko najdete na: http://ims.mf.uni-lj.si/archive/17(2)

3 Ocenjevanje v frekventisticni statistiki

Kako bi ocenili nas parameter s frekventisticno statistiko? Katere metode bi lahko uporabili?

Cenilka po metodi najvecjega verjetja in po metodi momentov je povprecje vzorca:

```
mean(x)
```

```
## [1] 1.57971
```

4 Ocenjevanje v Bayesovi statistiki

Bayesova formula:

$$\pi(\theta \mid x) \propto L(\theta \mid x)\, \pi(\theta).$$

4.1 Verjetje

Narisite verjetje tako, da bo ploscina pod narisano krivuljo enaka ena. Predrugacite lahko kodo, ki smo jo uporabili za binomski model.

$$L(\theta \mid x) = \prod_{i=1}^{n} \frac{1}{x_i!} \theta^{x_i} e^{-\theta} \propto \theta^{\sum_{i=1}^{n} x_i} e^{-n\theta}$$

Odvisnost od vzorca le preko $\sum_{i=1}^{n} x_i =: k$ (in n), ki sta na nasem vzorcu (ohranimo oznake kakor pri binomskem modelu):

```
(n <- length(x))
```

```
## [1] 69
```

```
(k <- sum(x))
```

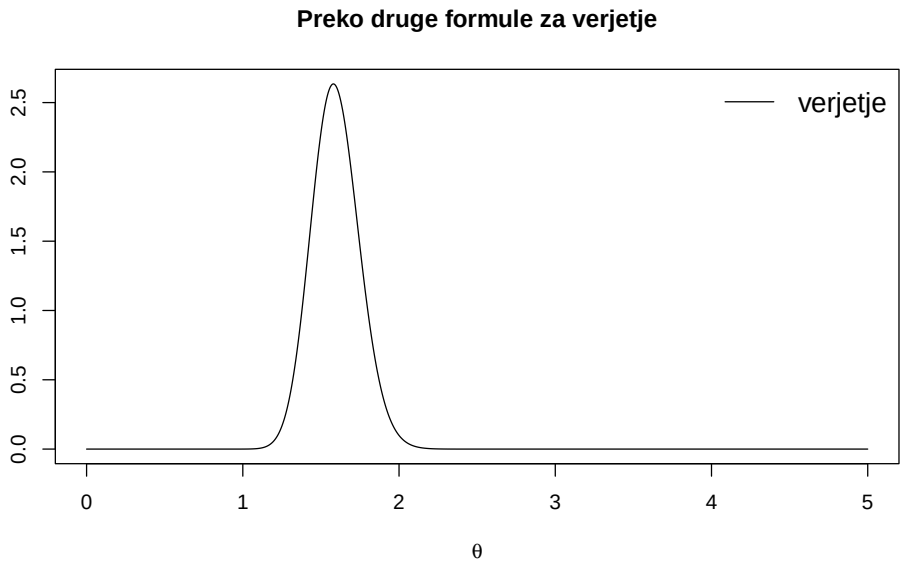
```
## [1] 109
```

V R-u:

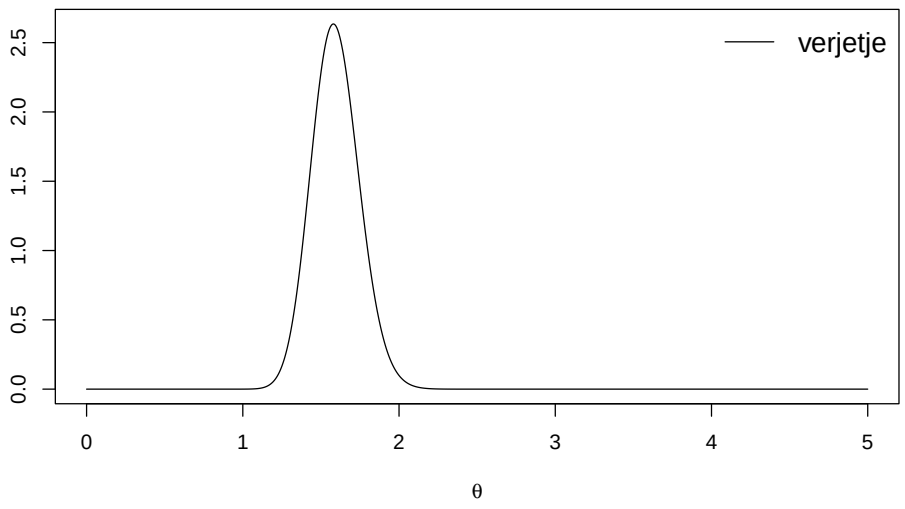
```
verjetje <- function(theta, k, n){
  theta^k * exp(-n*theta)
}

verjetje2 <- function(theta, x){
  prod(dpois(x, theta))
}

# POZOR: Ce verjetje implementiramo preko osnovne formule (glejte verjetje2),
# potem moramo biti pazlivejsi pri nadaljnji implementaciji,
# kjer uporabimo funkcijo verjetje2 na dveh vektorjih
# (nujno je uporabiti npr. sapply, glejte konst2 in kodo za spodnji graf).

#Z mnozenjem s konst dosezemo, da je integral verjetja glede na theta enak 1.
konst <- function(k, n){
  theta <- seq(0.001, 5, 0.001)
  1 / (0.001 * sum(verjetje(theta, k, n)))
}

konst2 <- function(x){
  theta <- seq(0.001, 5, 0.001)
  1 / (0.001 * sum(sapply(theta, verjetje2, x)))
}
```

Narisemo za nas vzorec:

```
theta <- seq(0, 5, 0.001)
konst.verjetje <- konst(k, n) * verjetje(theta, k, n)
plot(theta, konst.verjetje, type = "l",
     xlab = expression(theta), ylab ="", main = "Preko druge formule za verjetje")
legend("topright", legend = c("verjetje"), col = c("black"),
       lty = 1, bty = "n", cex = 1.3)
```




```
theta <- seq(0, 5, 0.001)
konst.verjetje2 <- konst2(x) * sapply(theta, verjetje2, x)
plot(theta, konst.verjetje2, type = "l",
     xlab = expression(theta), ylab ="", main = "Preko osnovne formule za verjetje")
legend("topright", legend = c("verjetje"), col = c("black"),
       lty = 1, bty = "n", cex = 1.3)
```



4.2 Apriorna porazdelitev

Za apriorno porazdelitev si izberemo gama porazdelitev, ki je v primeru poissonove porazdelitve podatkov *conjugate prior* (pomeni, da apriorna in aposteriorna porazdelitev pripadata enaki druzini porazdelitev), zato se lahko uporablja tudi izraz **gama-poissonov model**.

Za apriorno porazdelitev imamo torej gostoto gama porazdelitve pri parametrih $\alpha, \beta > 0$:

$$\pi(\theta) = \pi(\theta \mid \alpha, \beta) = \frac{\beta^\alpha}{\Gamma(\alpha)} \theta^{\alpha-1} e^{-\beta\theta},$$

kjer je funkcija gama $\Gamma(a) = (a-1)!$ za pozitivna cela stevila a. Spomnimo se:

- $\mathrm{E}(\mathrm{Gama}(\alpha, \beta)) = \frac{\alpha}{\beta}$,
- $\mathrm{var}(\mathrm{Gama}(\alpha, \beta)) = \frac{\alpha}{\beta^2}$.

Denimo, da se po nasih izkusnjah rodijo eni do dvoji cetvorcki letno, zato se odlocimo, da bo povprecje apriorne porazdelitve enako 1.5.

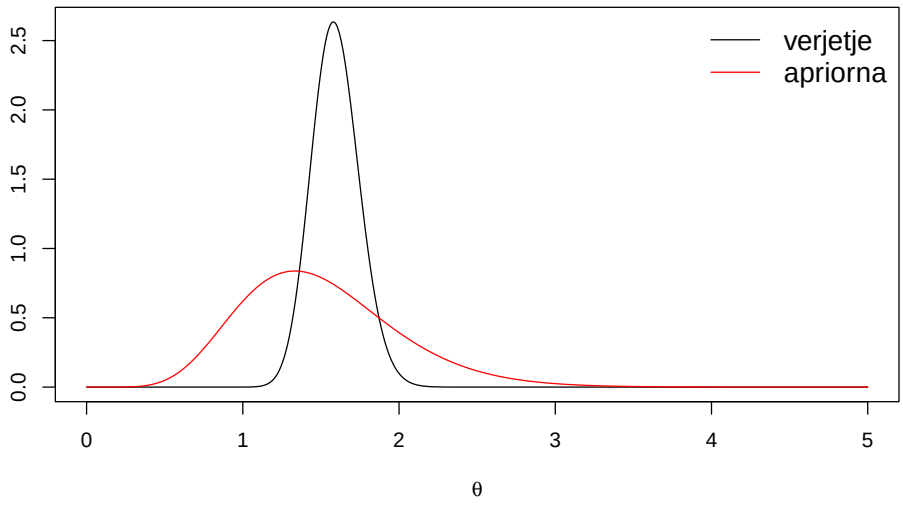
Pri tem mislimo, da se lahko zmotimo za priblizno 1. V primeru normalne porazdelitve je 95% vrednosti oddaljenih od povprecja za priblizno 2 standardna odklona. Standardni odklon nase porazdelitve zato nastavimo na 0.5.

Izracunajte α in β nase apriorne porazdelitve.

Na graf z verjetjem dodajte gostoto apriorne porazdelitve. Predrugacite lahko kodo, ki smo jo uporabili za binomski model.

Narisemo v R-u:

```
alpha <- 1.5*6
beta <- 6

theta <- seq(0, 5, 0.001)
apriorna <- dgamma(theta, shape = alpha, rate = beta)

konst.verjetje <- konst(k, n) * verjetje(theta, k, n)

y.max <- max(c(konst.verjetje, apriorna))
plot(theta, konst.verjetje, ylim = c(0, y.max), type = "l",
     xlab = expression(theta), ylab = "")
lines(theta, apriorna, col = "red")
legend("topright", legend = c("verjetje", "apriorna"), col = c("black", "red"),
       lty = 1, bty = "n", cex = 1.3)
```



4.3 Aposteriorna porazdelitev

Ker smo uporabili *conjugate prior*, bo aposteriorna porazdelitev tudi iz druzine gama porazdelitev. Koliksna sta njena parametra?

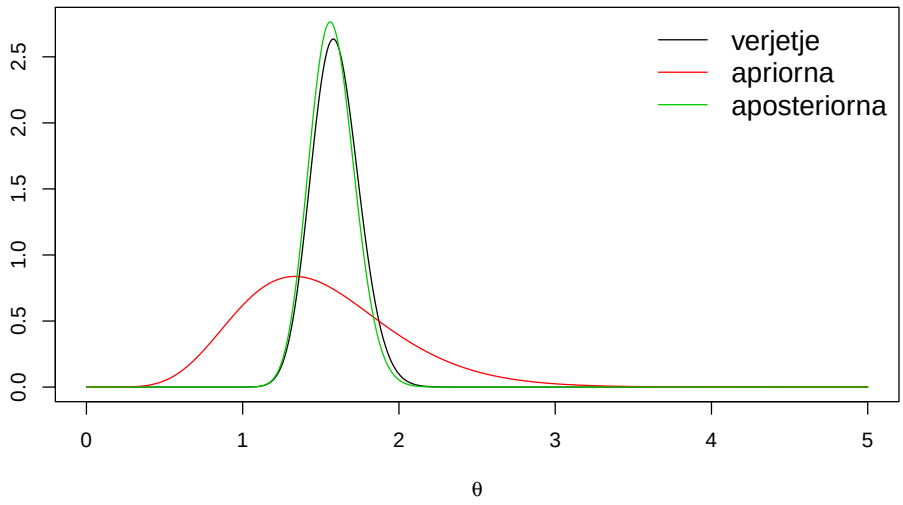
Na graf z verjetjem in gostoto apriorne porazdelitve dodajte se gostoto aposterirone porazdeitve. Predrugacite lahko kodo, ki smo jo uporabili za binomski model.

Njena parametra sta enaka:

- $\alpha_{\text{apost}} = k + \alpha$,
- $\beta_{\text{apost}} = n + \beta$.

Narisemo v R-u:

```
alpha.apost <- k + alpha
beta.apost <- n + beta

theta <- seq(0.001, 5, 0.001)
aposteriorna <- dgamma(theta, alpha.apost, beta.apost)

konst.verjetje <- konst(k, n) * verjetje(theta, k, n)
apriorna <- dgamma(theta, alpha, beta)

y.max <- max(c(konst.verjetje, apriorna, aposteriorna))
plot(theta, konst.verjetje, ylim=c(0, y.max), type = "l",
     xlab = expression(theta), ylab = "")
lines(theta, apriorna, col = "red")
lines(theta, aposteriorna, col = "green3")
legend("topright", legend = c("verjetje", "apriorna","aposteriorna"),
       col = c("black","red","green3"), lty = 1, bty = "n", cex = 1.3)
```



4.4 Ocena parametra θ

Ocenite parameter θ.

Ena moznost je pricakovana vrednost aposteriorne porazdelitve:

$$\hat{\theta} = \frac{\alpha_{\text{apost}}}{\beta_{\text{apost}}} = \frac{k+\alpha}{n+\beta}.$$

Podobno kot pri binomskem modelu lahko zapisemo

$$\hat{\theta} = \frac{\beta}{\beta+n} \cdot \mu + \frac{n}{\beta+n} \cdot \frac{k}{n},$$

kjer je $\mu = \text{E}(\text{Gama}(\alpha, \beta)) = \frac{\alpha}{\beta}$.

Ideja: $\hat{\theta}$ je utezeno povprecje med $\text{E}(\text{apriorna})$ in $\text{E}(X)$, kjer preko β kontroliramo, kako mocno verjamemo apriorni pricakovani vrednosti.

```
alpha.apost / beta.apost
```

```
## [1] 1.573333
```

4.5 Interval zaupanja

Izracunajte 95% interval zaupanja za θ. Preizkusite obe metodi, preko kvantilov in *highest posterior density (HPD) region*. Predrugacite lahko kodo, ki smo jo uporabili za binomski model.

Preko kvantilov porazdelitve:

```
(iz <- qgamma(c(0.025, 0.975), alpha.apost, beta.apost))
```

```
## [1] 1.302290 1.869619
```

Highest posterior density (HPD) region:

```
#install.packages("HDInterval")
library(HDInterval)

aposteriorna.sample <- rgamma(100000, alpha.apost, beta.apost)
(iz.hdi <- hdi(aposteriorna.sample, credMass = 0.95))
```

```
##    lower    upper
## 1.291588 1.859134
## attr(,"credMass")
## [1] 0.95
```

4.6 Testiranje hipotez

Kako verjetna je domneva, da se v povprecju na leto rodijo eni do dvoji cetvorcki?

Verjetnost te domneve je:

```
1 - pgamma(1, alpha.apost, beta.apost) -
    pgamma(2, alpha.apost, beta.apost, lower.tail = FALSE)
```

```
## [1] 0.9969729
```

4.7 Napovedovanje

Zanima nas, kaj lahko povemo o stevilu cetvorckov v prihajajocem letu ob upostevanju podatkov zadnjih 69 let, tj. zanima nas **aposteriorna napovedna porazdelitev**.

(Ce bi nas zanimalo stevilo cetvockov v prihajajocem letu brez upostevanja podatkov 69 let, potem bi nas zanimala **apriorna napovedna porazdelitev**.)

V Poissonovem modelu z apriorno gama porazdelitvijo lahko hitro izpeljemo (predavanja) apriorno/aposteriorno napovedno porazdelitev:

$$P(Y = K) = \frac{\Gamma(K + \tilde{\alpha})}{\Gamma(\tilde{\alpha})\, K!} \tilde{\beta}^{\tilde{\alpha}} / (\tilde{\beta} + 1)^{K+\tilde{\alpha}} \quad \text{za } K \in \{0, 1, 2, \ldots\}.$$

To je ravno negativna binomska porazdelitev s parametroma $r = \tilde{\alpha}$ in $p = 1/(1+\tilde{\beta})$, zasledimo pa lahko tudi poimenovanje **Gama-Poissonova porazdelitev**.

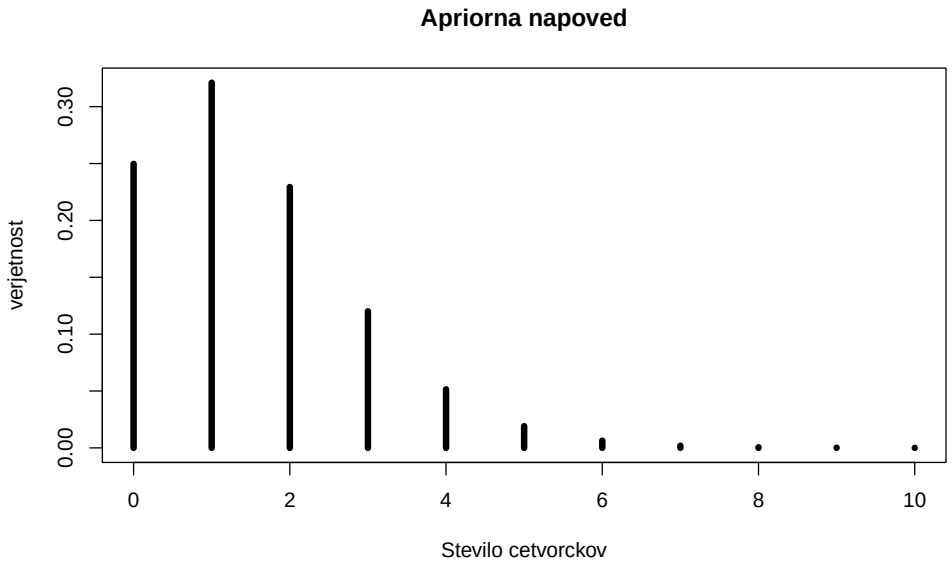
Za $\tilde{\alpha}, \tilde{\beta}$ vstavimo primerna parametra gama apriorne oz. aposteriorne porazdelitve.

Gama-Poissonova porazdelitev:

```
dgammapoiss <- function(K, a, b){
  gamma(K+a)/(gamma(a)*factorial(K)) * b^a / (b+1)^(K+a)
}
```

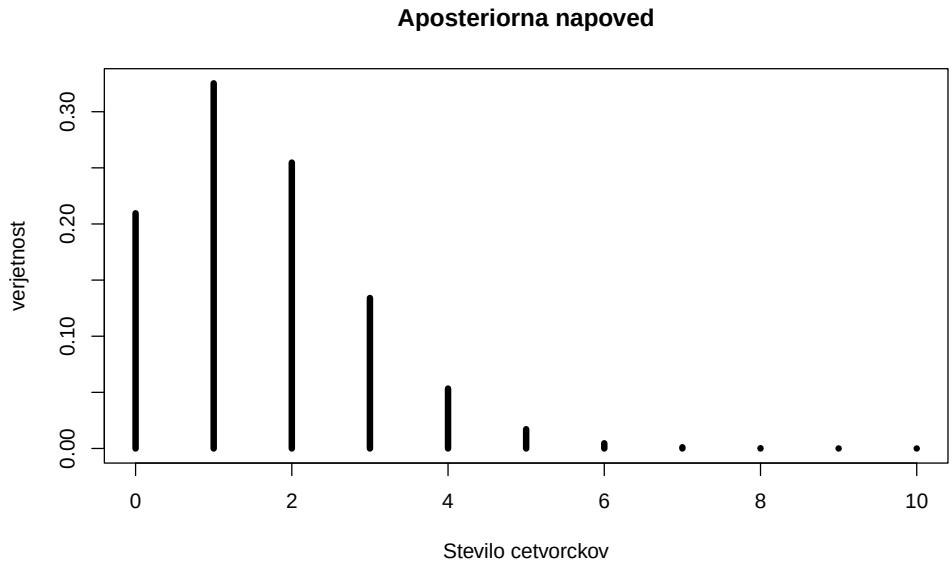
Narisemo apriorno napovedno porazdelitev.

```
plot(0:10, dgammapoiss(0:10, a = alpha, b = beta), type = "h",
     xlab = "Stevilo cetvorckov", ylab = "verjetnost",
     main = "Apriorna napoved", lwd = 5)
```



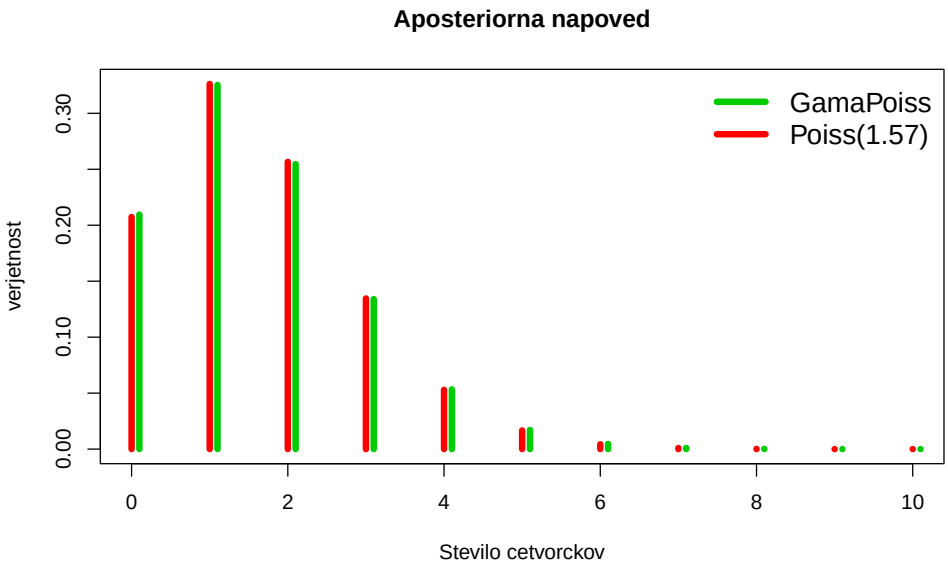
Narisemo aposteriorno napovedno porazdelitev.

```
plot(0:10, dgammapoiss(0:10, a = alpha.apost, b = beta.apost), type = "h",
     xlab = "Stevilo cetvorckov", ylab = "verjetnost",
     main = "Aposteriorna napoved", lwd = 5)
```



Poglejmo si se, kaksna je razlika med pravilno izracunano aposteriorno napovedno porazdelitvijo in tisto, ki jo dobimo, ce v Poissonovo porazdelitev vstavimo naso oceno parametra $\hat{\theta} = \alpha_{\text{apost}}/\beta_{\text{apost}} = 1.57$.

```
plot(0:10, dpois(0:10, alpha.apost / beta.apost), type = "h",
     xlab = "Stevilo cetvorckov", ylab = "verjetnost",
     main = "Aposteriorna napoved", col = "red", lwd = 5)
segments(x0 = seq(0.1,10.1,1), y0 = rep(0,11),
         x1 = seq(0.1,10.1,1), y1 = dgammapoiss(0:10, a = alpha.apost, b = beta.apost),
         lwd = 5, col = "green3")
legend("topright",  lty = 1, lwd = 5,
       c("GamaPoiss", paste("Poiss(",round(alpha.apost / beta.apost,2),")",sep="")),
       col = c("green3","red"), bty = "n", cex = 1.3)
```


Aposteriorna napoved
GamaPoiss
Poiss(1.57)
verjetnost
0.00
0.10
0.20
0.30
0
2
4
6
8
10
Stevilo cetvorckov

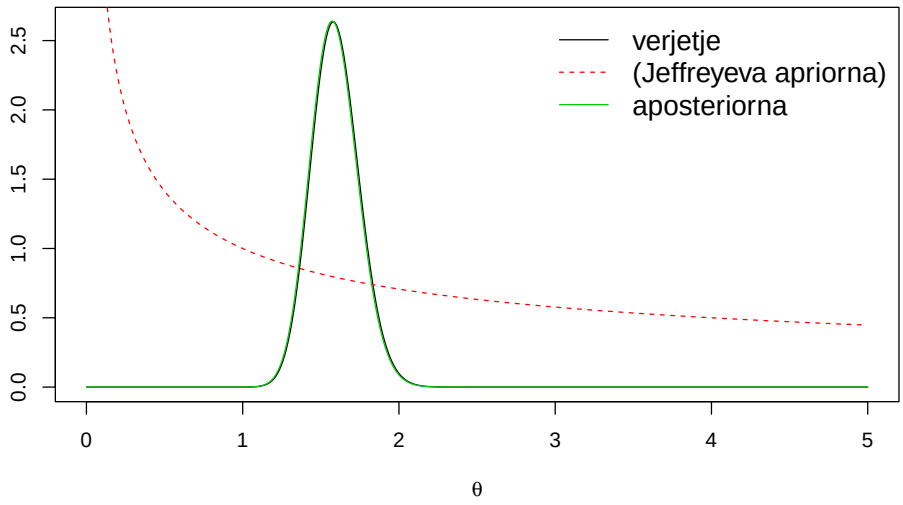
5 Jeffreyeva apriorna porazdelitev

Pri Poissonovem modelu je Jeffreyeva apriorna porazdelitev $\pi(\theta) \propto \sqrt{1/\theta}$, kar si lahko interpretiramo kakor gostoto Gama($\alpha = 0.5$, $\beta = 0$). Ker je $\int_0^\infty \sqrt{1/\theta}\, d\theta = \infty$, je to **improper prior**.

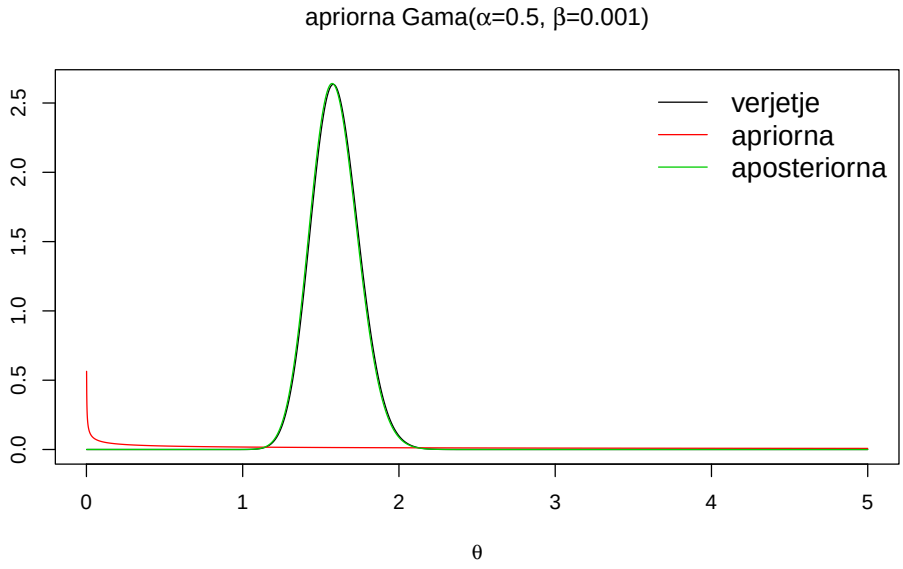
Pri taksni apriorni porazdelitvi bo aposteriorna porazdelitev Gama($\alpha_{\text{apost}} = k+0.5$, $\beta_{\text{apost}} = n$). **Nujno je, da je aposteriorna porazdelitev prava porazdelitev** (tj. integral gostote je enak 1), medtem ko to ne velja nujno za apriorno porazdelitev.

Pri nasih podatkih dobimo naslednje, kjer Jeffreyeve apriorne porazdelitve dejansko ne bi smeli narisati, saj njen integral ni enak 1.

```
alpha <- 0.5
beta <- 0

alpha.apost <- k + alpha
beta.apost <- n + beta

theta <- seq(0.001, 5, 0.001)
konst.verjetje <- konst(k, n) * verjetje(theta, k, n)
apriorna <- sqrt(1/theta)
aposteriorna <- dgamma(theta, alpha.apost, beta.apost)

plot(theta, konst.verjetje, type = "l",
     xlab = expression(theta), ylab = "")
lines(theta, apriorna, col = "red", lty = 2)
lines(theta, aposteriorna, col = "green3")
legend("topright", legend = c("verjetje", "(Jeffreyeva apriorna)","aposteriorna"),
       col = c("black","red","green3"), lty = c(1,2,1), bty = "n", cex = 1.3)
```



Spodnje dobimo, ce za parametra apriorne gama porazdelitve vzamemo $\alpha = 0.5$ in zelo majhen β, s cimer



3. sklop: Normalni model z znano varianco

1 Primer

Podan imamo naslednji vzorec visin (metri) studentov moskega spola:

```
x <- c(1.91, 1.94, 1.68, 1.75, 1.81, 1.83, 1.91, 1.95, 1.77, 1.98,
       1.81, 1.75, 1.89, 1.89, 1.83, 1.89, 1.99, 1.65, 1.82, 1.65,
       1.73, 1.73, 1.88, 1.81, 1.84, 1.83, 1.84, 1.72, 1.91, 1.63)
```

Zanima nas povprecna visina studentov, kjer privzamemo, da je standardni odklon $\sigma = 0.1$.

```
sigma <- 0.1
```

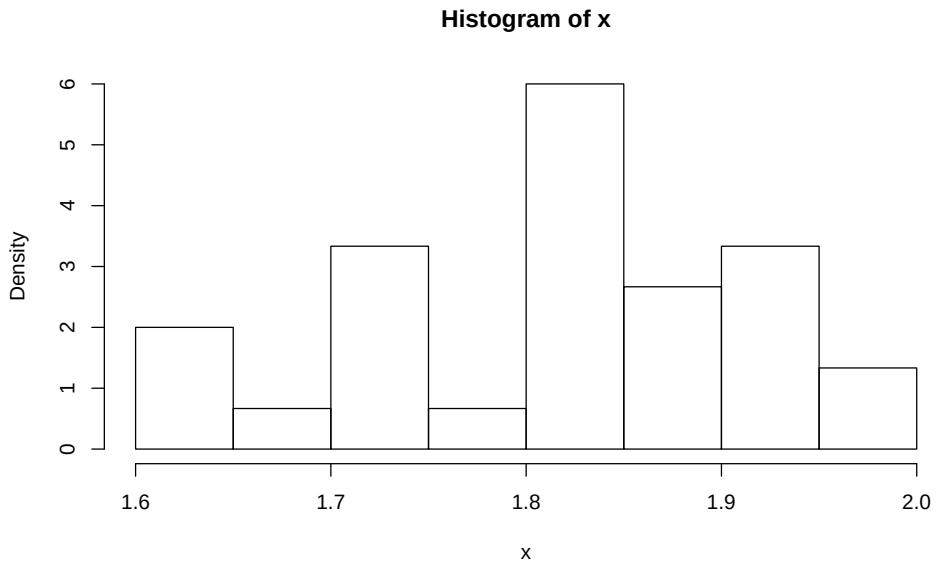
2 Verjetnostni model za nas primer

Vzorec $X_1, X_2, \ldots, X_n$, kjer je:

- $n = 30$ stevilo studentov,
- X_i predstavlja visino i-tega studenta,
- $X_i \mid \theta \sim N(\theta, \sigma^2 = 0.1^2)$,
- $f(x \mid \theta) = \frac{1}{\sqrt{2\pi}\,0.1} e^{-\frac{(x-\theta)^2}{2\cdot(0.1)^2}}$.

Ali je zgornji model smiseln za nase podatke?

```
hist(x, prob = TRUE)
```




```
shapiro.test(x)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  x
## W = 0.96666, p-value = 0.4523
```

```
sd(x)
```

```
## [1] 0.09857374
```

3 Ocenjevanje v frekventisticni statistiki

Cenilka po metodi najvecjega verjetja in po metodi momentov je povprecje vzorca:

```
mean(x)
```

```
## [1] 1.820667
```

4 Ocenjevanje v Bayesovi statistiki

Bayesova formula:

$$\pi(\theta \mid x) \propto L(\theta \mid x)\, \pi(\theta).$$

4.1 Verjetje

Narisite verjetje tako, da bo ploscina pod narisano krivuljo enaka ena.

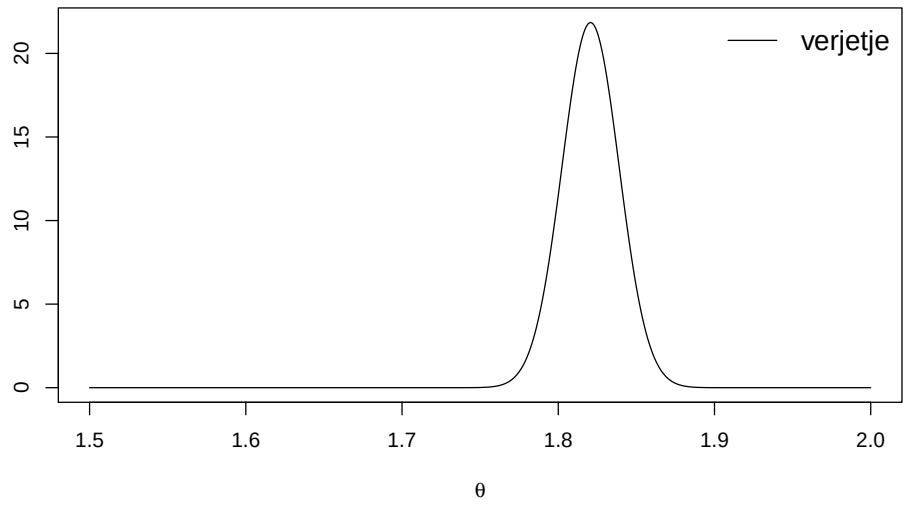
$$L(\theta \mid x) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\,0.1} e^{-\frac{(x_i-\theta)^2}{2\cdot(0.1)^2}}$$

V R-u:

```
verjetje <- function(theta, x, sigma = 0.1){
  prod(dnorm(x, mean = theta, sd = sigma))
}

#Z mnozenjem s konst dosezemo, da je integral verjetja glede na theta enak 1.
konst <- function(x, from = 1.5, to = 2, by = 0.001, sigma = 0.1){
  theta <- seq(from = from, to = to, by = by)
  1 / (by * sum(sapply(theta, FUN = verjetje, x = x, sigma = sigma)))
}
```

Narisemo za nas vzorec:

```
theta <- seq(1.5, 2, 0.001)
konst.verjetje <- konst(x) * sapply(theta, FUN = verjetje, x = x, sigma = sigma)
plot(theta, konst.verjetje, type = "l",
     xlab = expression(theta), ylab ="")
legend("topright", legend = c("verjetje"), col = c("black"),
       lty = 1, bty = "n", cex = 1.3)
```



4.2 Apriorna porazdelitev

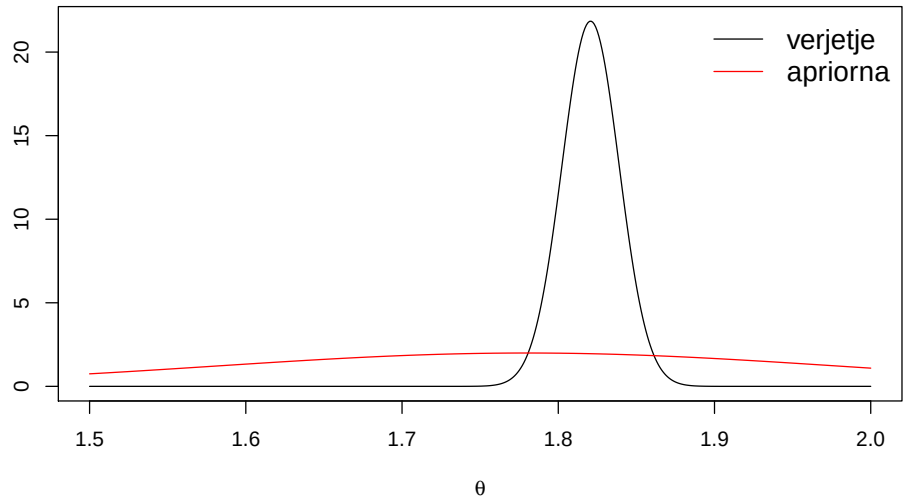
V tem modelu je konjugirana porazdelitev normalna porazdelitev, njena parametra bomo oznacili z μ_0 in σ_0^2.

Jeffrejeva apriorna porazdelitev v tem modelu je $\pi(\theta) \propto \sqrt{1/\sigma^2} \propto 1$, kar si lahko interpretiramo kakor gostoto $N(\mu_0 = 0, \sigma_0^2 =$ "zelo velik"). Ker je $\int_{-\infty}^{\infty} 1 \, d\theta = \infty$, je to *improper prior*.

Na spletnih straneh SURS-a (Statisticni urad republike Slovenije) lahko najdemo podatek, da je povprecna visina moskih 178 cm (leto 2015), zaradi cesar se odlocimo za $\mu_0 = 1.78$. Odlocimo se za $\sigma_0^2 = 0.2^2$, tj. 95% referencni interval apriorne porazdelitve bo priblizno 178 cm $\pm$ 40 cm oz. [138 cm, 218 cm] (sibko informativna porazdelitev).

Narisemo v R-u:

```
mu0 <- 1.78
sigma0 <- 0.2

theta <- seq(1.5, 2, 0.001)
konst.verjetje <- konst(x) * sapply(theta, FUN = verjetje, x = x, sigma = sigma)
apriorna <- dnorm(theta, mean = mu0, sd = sigma0)

y.max <- max(c(konst.verjetje, apriorna))
plot(theta, konst.verjetje, ylim = c(0, y.max), type = "l",
     xlab = expression(theta), ylab = "")
lines(theta, apriorna, col = "red")
legend("topright", legend = c("verjetje", "apriorna"), col = c("black", "red"),
       lty = 1, bty = "n", cex = 1.3)
```



4.3 Aposteriorna porazdelitev

Ker smo uporabili konjugirano porazdelitev, bo tudi aposteriorna porazdelitev normalna.

Njena parametra, ki ju oznacimo z μ_n in σ_n^2, sta enaka:

$$\frac{1}{\sigma_n^2} = \frac{1}{\sigma_0^2} + \frac{n}{\sigma^2},$$

$$\mu_n = \frac{1/\sigma_0^2}{1/\sigma_0^2 + n/\sigma^2}\,\mu_0 + \frac{n/\sigma^2}{1/\sigma_0^2 + n/\sigma^2}\,\bar{x},$$

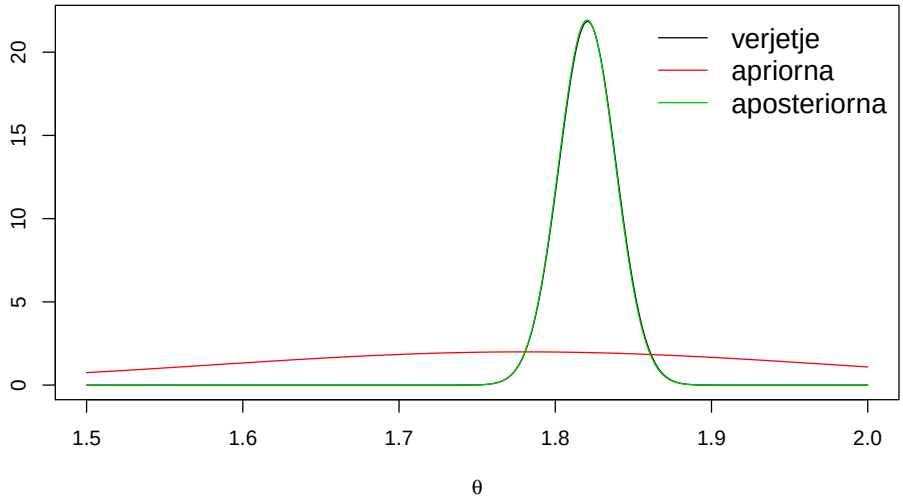
kjer je $\sigma = 0.01$.

Aposteriorna pricakovana vrednost μ_n je torej utezeno povprecje apriorne pricakovane vrednosti μ_0 in vzorcnega povprecja $\bar{x}$, kjer preko *precision* apriorne porazdelitve $1/\sigma_0^2$ kontroliramo, kako mocno verjamemo apriorni pricakovani vrednosti.

V primeru Jeffrejeve apriorne porazdelitve dobimo $\mu_n = \bar{x}$ in $\sigma_n^2 = \sigma^2/n$. Ali je to skladno s frekventisticno statistiko? Zakaj?

Narisemo v R-u:

```
n <- length(x)
prec <- 1/sigma^2
prec0 <- 1/sigma0^2

prec.n <- prec0 + n*prec
sigma.n <- sqrt(1/prec.n)

mu.n <- prec0/prec.n * mu0 + n*prec/prec.n * mean(x)

theta <- seq(1.5, 2, 0.001)
konst.verjetje <- konst(x) * sapply(theta, FUN = verjetje, x = x, sigma = sigma)
apriorna <- dnorm(theta, mean = mu0, sd = sigma0)
aposteriorna <- dnorm(theta, mean = mu.n, sd = sigma.n)

y.max <- max(c(konst.verjetje, apriorna, aposteriorna))
plot(theta, konst.verjetje, ylim=c(0, y.max), type = "l",
     xlab = expression(theta), ylab = "")
lines(theta, apriorna, col = "red")
lines(theta, aposteriorna, col = "green3")
legend("topright", legend = c("verjetje", "apriorna","aposteriorna"),
       col = c("black","red","green3"), lty = 1, bty = "n", cex = 1.3)
```



4.4 Ocena parametra θ

Ocenimo parameter θ s pricakovano vrednostjo aposteriorne porazdelitve:

$$\hat{\theta} = \mu_n.$$

```
mu.n
```

```
## [1] 1.820331
```

4.5 Interval zaupanja

Izracunamo 95% interval zaupanja za θ.

Preko kvantilov porazdelitve:

```
(iz <- qnorm(c(0.025, 0.975), mean = mu.n, sd = sigma.n))
```

```
## [1] 1.784695 1.855966
```

Highest posterior density (HPD) region:

```
#install.packages("HDInterval")
library(HDInterval)

aposteriorna.sample <- rnorm(1000000, mean = mu.n, sd = sigma.n)
(iz.hdi <- hdi(aposteriorna.sample, credMass = 0.95))
```

```
##    lower    upper
## 1.784735 1.855984
## attr(,"credMass")
## [1] 0.95
```

Katera metoda se vam zdi pri tem modelu boljsa? Zakaj?

4.6 Napovedovanje

Zanima nas, kaj lahko povemo o visini novega studenta ob upostevanju podatkov 30 studentov, tj. zanima nas **aposteriorna napovedna porazdelitev**.

(Ce bi nas zanimala visina studenta brez upostevanja podatkov 30 studentov, potem bi nas zanimala **apriorna napovedna porazdelitev**.)

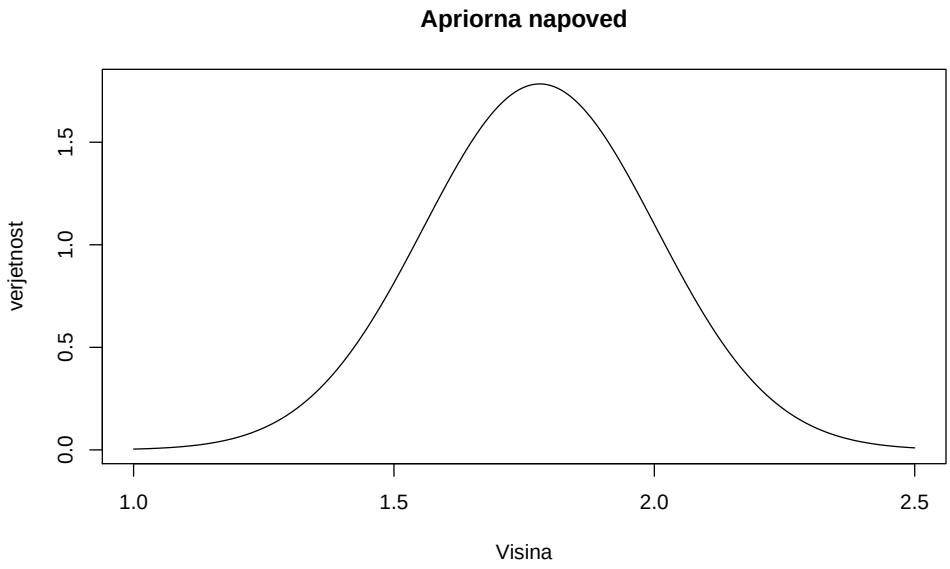
V tem modelu je apriorna/aposteriorna napovedna porazdelitev normalna z naslednjimi parametri:

- apriorna napovedna porazdelitev: povprecje μ_0, varianca $\sigma_0^2 + \sigma^2$,
- aposteriorna napovedna porazdelitev: povprecje μ_n, varianca $\sigma_n^2 + \sigma^2$.

Ne glede na to, kako velik vzorec imamo oz. kako natancna je nasa aposterorna porazdelitev (majhen σ_n^2), bo varianca aposteriorne napovedne porazdelitve vsaj σ^2.

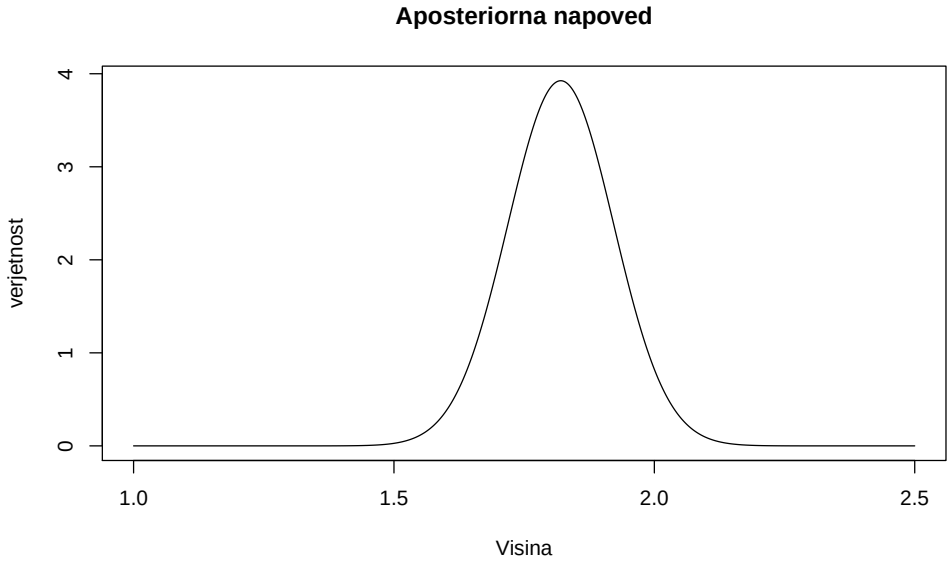
Narisemo apriorno napovedno porazdelitev.

```
theta <- seq(1, 2.5, 0.001)
plot(theta, dnorm(theta, mean = mu0, sd = sqrt(sigma0^2 + sigma^2)), type = "l",
     xlab = "Visina", ylab = "verjetnost",
     main = "Apriorna napoved")
```



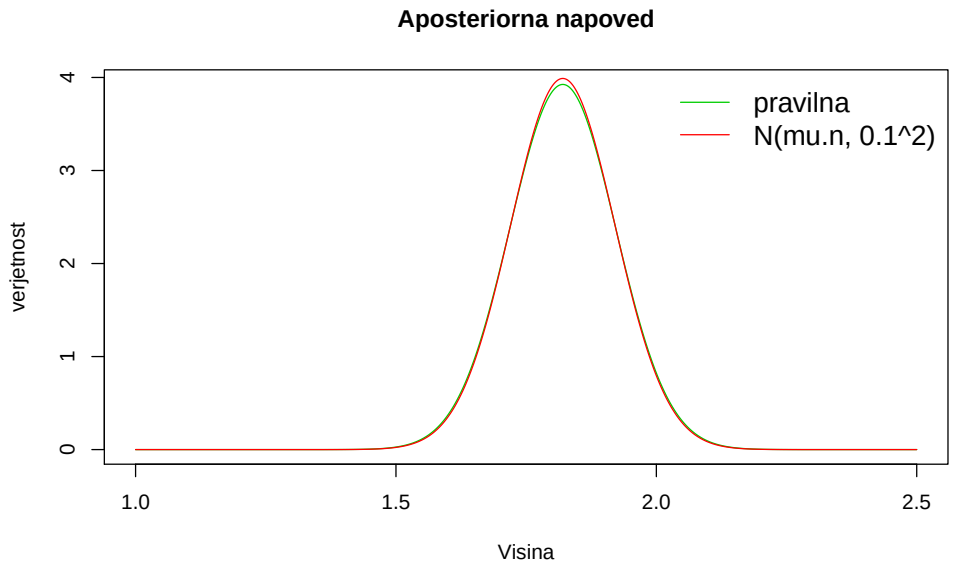
Narisemo aposteriorno napovedno porazdelitev.

```
theta <- seq(1, 2.5, 0.001)
plot(theta, dnorm(theta, mean = mu.n, sd = sqrt(sigma.n^2 + sigma^2)), type = "l",
     xlab = "Visina", ylab = "verjetnost",
     main = "Aposteriorna napoved")
```



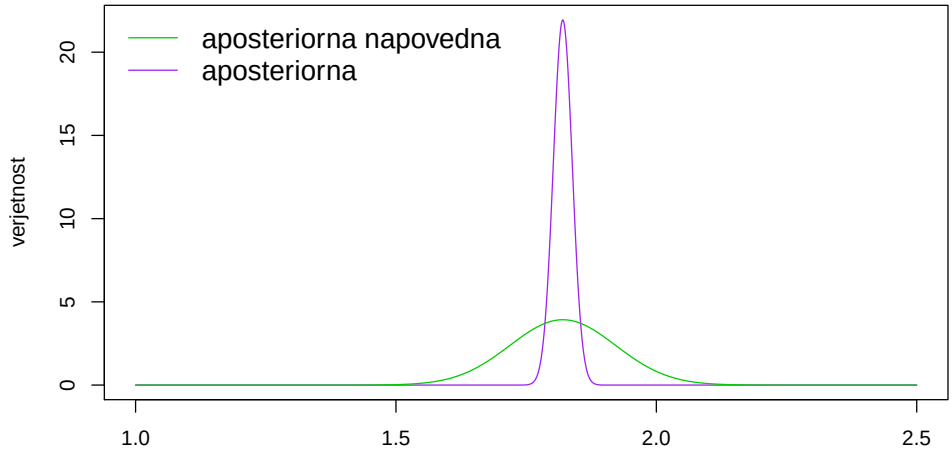
Poglejmo si se, kaksna je razlika med pravilno izracunano aposteriorno napovedno porazdelitvijo in tisto, ki jo dobimo, ce v normalno porazdelitev z znano varianco $\sigma^2 = 0.1^2$ vstavimo naso oceno parametra $\hat{\theta} = \mu_n$, torej primerjamo s porazdelitvijo $N(\mu_n, \sigma^2)$.

```
theta <- seq(1, 2.5, 0.001)
plot(theta, dnorm(theta, mean = mu.n, sd = sqrt(sigma.n^2 + sigma^2)), type = "l",
     xlab = "Visina", ylab = "verjetnost",
     main = "Aposteriorna napoved", col="green3")
lines(theta, dnorm(theta, mean = mu.n, sd = sigma), col = "red")
legend("topright",  lty = 1,
       c("pravilna", "N(mu.n, 0.1^2)"), col = c("green3","red"), bty = "n", cex = 1.3)
```



Poudarimo bistveno razliko med aposteriorno porazdelitvijo povprecne visine in aposteriorno napovedno porazdelitvijo za visino novega studenta:

```
theta <- seq(1, 2.5, 0.001)
plot(theta, dnorm(theta, mean = mu.n, sd = sigma.n), type = "l",
     xlab = "", ylab = "verjetnost", col="purple")
lines(theta, dnorm(theta, mean = mu.n, sd = sqrt(sigma.n^2 + sigma^2)), col="green3")
legend("topleft",  lty = 1,
       c("aposteriorna napovedna", "aposteriorna"), col = c("green3","purple"),
       bty = "n", cex = 1.3)
```



4. sklop: Algoritem Metropolis-Hastings

Imamo nek model, za katerega poznamo verjetje $L(\theta \mid x)$ (θ je lahko vektor parametrov, podatki x pa matrika). Odlocimo se za apriorno porazdelitev $\pi(\theta)$, njeno formulo torej poznamo.

Izracunamo torej *le se* aposteriorno porazdelitev $\pi(\theta \mid x)$ preko Bayesove formule

$$\pi(\theta \mid x) \propto L(\theta \mid x)\, \pi(\theta).$$

Preko aposteriorne porazdelitve izracunamo *vse, kar nam srce pozeli*: za tockovno oceno vzamemo povprecje ali pa mediano, za interval zaupanja izracunamo primerne kvantile, izracunamo verjetnosti nasih domnev.

Kaj je tu problem?

1 MCMC (Markov Chain Monte Carlo)

Zelimo aposteriorno porazdelitev za parameter θ, natancneje vzorec iz aposteriorne porazdelitve.

Ideja algoritma:

1. Izberemo si zacetno vrednost $\theta^{(0)}$.
2. *Najdemo primerno* porazdelitev $p(\cdot|\cdot)$, tako da bomo lahko na vsakem koraku i dobili $\theta^{(i+1)}$ kot simulacijo iz porazdelitve $p(\theta^{(i+1)}|\theta^{(i)})$.
3. Po n korakih dobimo realizacijo $\theta^{(0)}, \theta^{(1)}, \ldots, \theta^{(n)}$. Postopek ponavljamo *dokler ne dosezemo zeljene konvergence*, pri cemer v koncnem vzorcu izpustimo zacetnih nekaj vrednosti (*burn-in*, npr. m).
4. Izbrani vzorec $\theta^{(m+1)}, \theta^{(m+2)}, \ldots, \theta^{(n)}$ je vzorec iz aposteriorne porazdelitve.
5. Preizkusimo razlicne zacetne vrednosti in primerjamo dobljene rezultate.

(6. Preizkusimo razlicne apriorne porazdelitve in primerjamo dobljene rezultate.)

2 Algoritem Metropolis-Hastings

Algoritem Metropolis-Hastings je eden izmed MCMC algoritmov.

Najprej si izberemo predlagalno porazdelitev (*proposal distribution*) $q(\cdot|\cdot)$, ki je *primerne* oblike.

En korak Metropolis-Hastings algoritma, ko imamo iz prejsnjega koraka na voljo $\theta^{(i)}$:

1. Simuliramo kandidata θ^* iz porazdelitve $q(\cdot|\theta^{(i)})$.
2. Izracunamo verjetnost sprejetja (*acceptance probability*)

$$\alpha = \min\left\{1, \frac{L(\theta^*|x)\ \pi(\theta^*)\ \ q(\theta^{(i)}|\theta^*)}{L(\theta^{(i)}|x)\pi(\theta^{(i)})q(\theta^*|\theta^{(i)})}\right\}.$$

3. Simuliramo u iz enakomerne zvezne porazdelitve na [0,1].
4. Ce je $u \leq \alpha$, izberemo kandidata (*accept*) in postavimo $\theta^{(i+1)} = \theta^*$.
   Ce je $u > \alpha$, zavrnemo kandidata (*reject*) in postavimo $\theta^{(i+1)} = \theta^{(i)}$.

3 Algoritem Metropolis-Hastings za primer normalnega modela z znano varianco

Uporabili bomo algoritem Metropolis-Hastings za primer iz 3. sklopa, kjer so bili nasi podatki vzorec visin (metri) studentov moskega spola:

```
x <- c(1.91, 1.94, 1.68, 1.75, 1.81, 1.83, 1.91, 1.95, 1.77, 1.98,
       1.81, 1.75, 1.89, 1.89, 1.83, 1.89, 1.99, 1.65, 1.82, 1.65,
       1.73, 1.73, 1.88, 1.81, 1.84, 1.83, 1.84, 1.72, 1.91, 1.63)
```

Privzeli smo normalni model $N(\theta, \sigma^2 = 0.1^2)$ in zeleli oceniti θ.

Verjetje tega modela je enako

$$L(\theta \mid x) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\,0.1} e^{-\frac{(x_i-\theta)^2}{2\cdot(0.1)^2}}.$$

Za apriorno porazdelitev smo si izbrali normalno porazdelitev (konjugirana v tem modelu) s povprecjem $\mu_0 = 1.78$ in varianco $\sigma_0^2 = 0.2^2$.

Za aposteriorno porazdelitev smo zato dobili normalno porazdelitev s parametroma μ_n in σ_n^2, kjer je

$$\frac{1}{\sigma_n^2} = \frac{1}{\sigma_0^2} + \frac{n}{\sigma^2},$$

$$\mu_n = \frac{1/\sigma_0^2}{1/\sigma_0^2 + n/\sigma^2}\,\mu_0 + \frac{n/\sigma^2}{1/\sigma_0^2 + n/\sigma^2}\,\bar{x}.$$

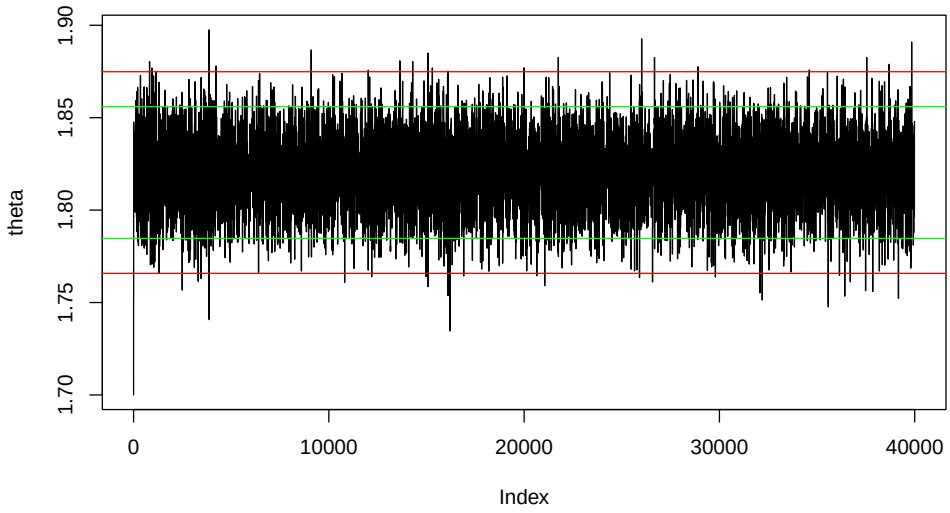
(Zapisemo lahko tudi preko *precision*=1/varianca.)

Pravo aposteriorno porazdelitev torej poznamo.

Sedaj jo bomo aproksimirali s pomocjo Metropolis-Hastings algoritma.

Najprej si moramo smiselno izbrati predlagalno porazdelitev $q(\cdot|\theta^{(i)})$. Katero bi si izbrali? Normalno s povprecjem $\theta^{(i)}$ in nekim standardnim odklonom.

Primer zaporedja iz aposteriorne porazdelitve, ki *izgleda dobro*:



Na sliki smo za boljso predstavo oznacili 95% referencni interval prave aposterirone porazdelitve (zeleni crti) in odmik od povprecja aposteriorne porazdelitve za 3 standardne odklone, tj. 99.7% referencni interval (rdeci crti). **Pozor:** S tem smo uporabili vedenje o pravi aposteriorni porazdelitvi, ki v realni situaciji ni znana (ravno zato jo z MCMC metodami tudi ocenjujemo).

Pomembno: Cilj konvergence je porazdelitev, ne pa stevilo.

3.1 Naloga

Za primer iz 3. sklopa (uporabite zgornje podatke, model z $\sigma^2 = 0.1^2$ in zgornjo apriorno porazdelitev z $\mu_0 = 1.78$ in $\sigma_0^2 = 0.2^2$ – ti parametri so fiksni) a proksimirajte aposteriorno porazdelitev s pomocjo algoritma Metropolis-Hastings, kjer sledite spodnjim korakom.

1. Sami v R-u sprogramirajte algoritem Metropolis-Hastings za primer ocenjevanja enega parametra oz. za nas primer. Kljucno je, da ga sprogramirate sami, pri cemer splosnost kode in efektivnost implementacije nista pomembni. (Za ta preprost primer boste npr. 40000 iteracij dobili v zelo kratkem casu, ne glede na izbor parametrov v spodnjih tockah ali efektivnost implementacije.)
2. Preizkusite ga na nasem primeru, kjer si sami izberite neko smiselno zacetno vrednost in varianco predlagalne porazdelitve. Rezultate predstavite na naslednji nacin:
   - Narisite celotno dobljeno zaporedje $\theta^{(i)}$ (glede na iteracije i).
   - Narisite le prvih 500 ali pa 5000 clenov.
   - Narisite celotno zaporedje, kjer uporabite ustrezen *burn-in*.
   - Za tako izbrano zaporedje graficno predstavite aposteriorno porazdelitev in jo graficno primerjajte s pravo aposteriorno porazdelitvijo.
   - Ocenite parameter in 95% interval zaupanja za parameter iz izbranega zaporedja ter primerjajte z ocenami iz prave aposterirone porazdelitve.
3. Pozenite vas algoritem pri neki nesmiselni zacetni vrednosti. **Pozor:** ce boste α implementirali po formuli iz str. 2, potem algoritem pri zelo nesmiselnih zacetnih vrednostih ne bo deloval – zato je potrebno implementirati na ravni logaritma (primerno prilagodite korake algoritma). Rezultate predstavite:
   - Za visje tocke domace naloge: Opisite, zakaj konkretno so se pojavile tezave, ce ste uporabili zelo nesmiselno zacetno vrednost in osnovno verzijo algoritma (brez logaritmiranja). Algoritem ustrezno implementirajte tako, da bo deloval tudi pri zelo nesmiselnih zacetnih vrednostih.
   - Ce zgornje ne uspete narediti, pustite algoritem v osnovni varianti in izberite nekoliko manj nesmiselno zacetno vrednost.
   - Narisite celotno dobljeno zaporedje $\theta^{(i)}$ (glede na iteracije i).
   - Narisite le prvih 500 ali pa 5000 clenov.
   - Narisite celotno zaporedje, kjer uporabite ustrezen *burn-in*.
4. Pri neki smiselni zacetni vrednosti pozenite algoritem pri nekaj razlicnih variancah za predlagalno porazdelitev. Pri izboru pretiravajte v obe smeri (spomnite se, kaksni so po velikosti nasi podatki), tako da boste graficno opazili razlike na prvih npr. 500 iteracijah. Rezultate predstavite:
   - Za vsak primer narisite prvih nekaj (nekje med 500 in 5000) clenov in se celotno zaporedje.
   - Komentirajte razlike in zakaj do njih pride. Kaj in zakaj vas moti pri izbranih primerih?
   - **Bonus vprasanje (ne steje v osnovne tocke domace naloge):** Kaksen bi bil v splosnem (ne vezano na nas vzorec) vas predlog glede izbora variance predlagalne porazdelitve oz. kaksen bi bil predlog za izbor koncnega zaporedja?

3.2 Resitve

Fiksni parametri nasega modela:

```
sigma <- 0.1

mu0 <- 1.78
sigma0 <- 0.2
```

Parametri prave aposteriorne porazdelitve:

```
n <- length(x)
prec <- 1/sigma^2
prec0 <- 1/sigma0^2

prec.n <- prec0 + n*prec
sigma.n <- sqrt(1/prec.n)

mu.n <- prec0/prec.n * mu0 + n*prec/prec.n * mean(x)
```

Porazdelitve, ki bodo nastopale v algoritmu Metropolis-Hastings (podane izven algoritma) – implementacija na logaritemski skali (najbolj preprosto uporabiti `log = TRUE`):

```
#Logaritem verjetja
loglik <- function(x, theta, sigma = sigma) {
   sum(dnorm(x, mean = theta, sd = sigma, log = TRUE))
}

#Apriorna porazdelitev (log)
logprior <- function(theta, mu0 = mu0, sigma0 = sigma0) {
  dnorm(theta, mean = mu0, sd = sigma0, log = TRUE)
}

#Predlagalna porazdelitev - vzorcenje
rq <- function(theta, sigma.q) {
  rnorm(1, mean = theta, sd = sigma.q)
}

#Predlagalna porazdelitev - gostota (log)
logdq <- function(theta.arg, theta.cond, sigma.q) {
  dnorm(theta.arg, mean = theta.cond, sd = sigma.q, log = TRUE)
}
```

Algoritem Metropolis-Hastings:

```
mh <- function(n.iter=40500, theta.init=1.7, sigma.q=0.1,
               x=x, sigma=sigma, mu0 = mu0, sigma0 = sigma0) {
  theta <- rep(NA, n.iter)
  theta[1] <- theta.init

  for(i in 2:n.iter) {
    new.theta <- rq(theta[i - 1], sigma.q=sigma.q)

    #Log-Accepance probability
    logacc.prob <- loglik(x, theta=new.theta, sigma=sigma) +
      logprior(theta=new.theta, mu0 = mu0, sigma0 = sigma0) +
      logdq(theta[i - 1], new.theta, sigma.q=sigma.q) -
      loglik(x, theta=theta[i - 1], sigma=sigma) -
      logprior(theta=theta[i - 1], mu0 = mu0, sigma0 = sigma0) -
      logdq(new.theta, theta[i - 1], sigma.q=sigma.q)
    logacc.prob <- min(0, logacc.prob) #0 = log(1)

    if(log(runif(1)) < logacc.prob) {
      #Accept
      theta[i] <- new.theta
    } else {
      #Reject
      theta[i] <- theta[i - 1]
    }
  }
  return(theta)
}
```

Zakaj smo ga implementirali na logaritemski skali?

Ob nesmiselni zacetni vrednosti bi bili vrednosti verjetja in apriorne porazdelitve v zacetni vrednosti in naslednjem kandidatu tako majhne, da bi dobili v stevcu in imenovalcu 0 in s tem *acceptance probability* ne bi bil definiran (NaN). Preizkusimo:

```
#Verjetje
lik <- function(x, theta, sigma = sigma) {
   sum(dnorm(x, mean = theta, sd = sigma))
}

#Apriorna porazdelitev
prior <- function(theta, mu0 = mu0, sigma0 = sigma0) {
  dnorm(theta, mean = mu0, sd = sigma0)
}
```

```r
#Predlagalna porazdelitev - gostota
dq <- function(theta.arg, theta.cond, sigma.q) {
  dnorm(theta.arg, mean = theta.cond, sd = sigma.q)
}

theta.init=10
sigma.q=0.1
n.iter=40500

theta <- rep(NA, n.iter)
theta[1] <- theta.init

i = 2
new.theta <- rq(theta[i - 1], sigma.q=sigma.q)

(stevec <- lik(x, theta=new.theta, sigma=sigma)*
   prior(theta=new.theta, mu0 = mu0, sigma0 = sigma0)*
   dq(theta[i - 1], new.theta, sigma.q=sigma.q))
```

```
## [1] 0
```

```r
(imenovalec <- lik(x, theta=theta[i - 1], sigma=sigma)*
   prior(theta=theta[i - 1], mu0 = mu0, sigma0 = sigma0)*
   dq(new.theta, theta[i - 1], sigma.q=sigma.q))
```

```
## [1] 0
```

```r
(acc.prob <- stevec/imenovalec)
```

```
## [1] NaN
```

Ali bi lahko pri izbrani predlagalni porazdelitvi algoritem poenostavili?

Zaradi simetricnosti normalne porazdelitve je $q(\theta^{(i)}|\theta^*) = q(\theta^*|\theta^{(i)})$, ta clen bi torej lahko pri tem izboru vrste predlagalne porazdelitve izpustili. Preizkusimo:

```r
dq(1.7, 1.8, sigma.q)
```

```
## [1] 2.419707
```

```r
dq(1.8, 1.7, sigma.q)
```

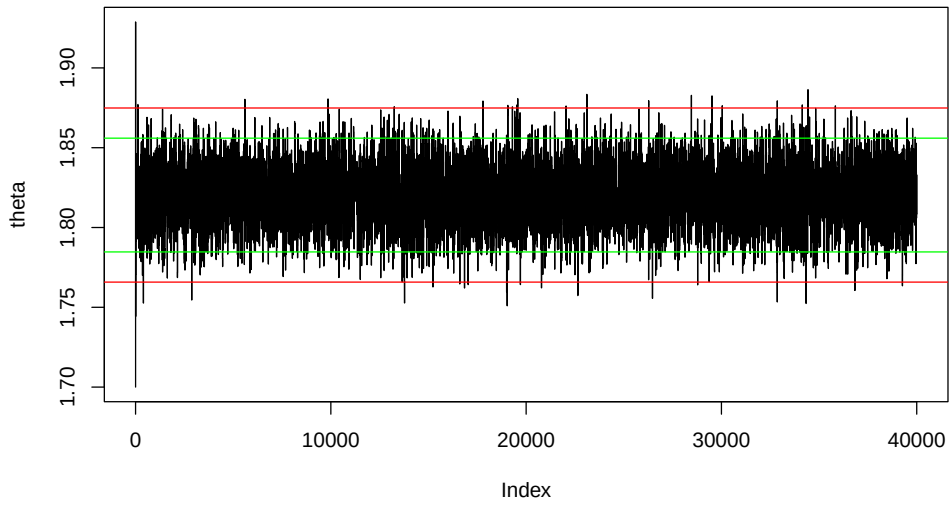
```
## [1] 2.419707
```

3.2.1 Primer s smiselno zacetno vrednostjo in "ustrezno" varianco predlagalne porazdelitve

Uporabimo zacetno vrednost 1.7 metra (smiselno za visino, povprecje vzorca 1.8) in standardni odklon (SO) predlagalne porazdelitve 0.1. V naslednjem koraku se bomo torej lahko oddaljili najvec do nekje $\pm 3 \cdot 0.1 = \pm 0.3$ metra. To je po eni strani gotovo dovolj, da raziscemo celotno obmocje aposteriorne porazdelitve (razpon vzorca od 1.6 do 2, ocenjujemo povprecje), po drugi strani pa nas ne omeji na premajhno obmocje okoli prejsnje vrednosti $\theta^{(i)}$. Seveda bi bila tudi marsikatera druga vrednost za SO predlagalne porazdelitve ustrezna.

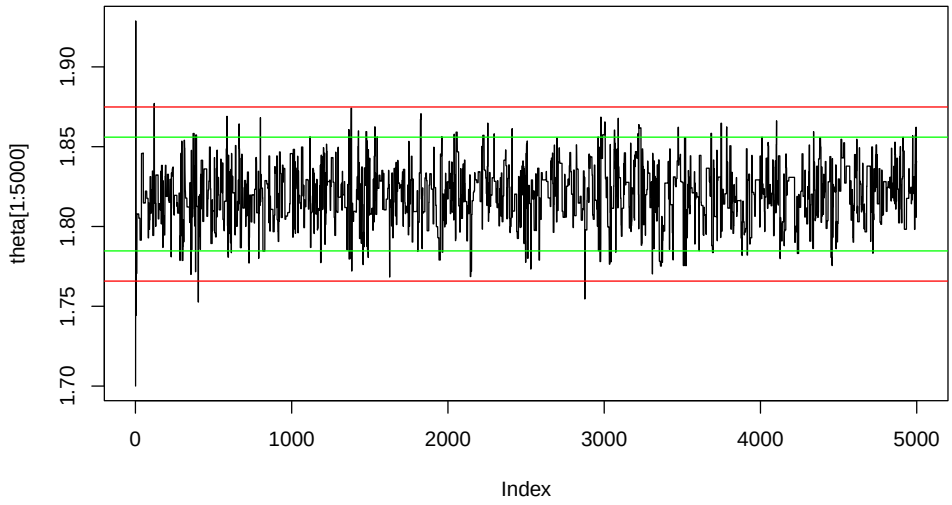
Narisemo dobljeni vzorec:

```
theta <- mh(n.iter=40000, theta.init=1.7, sigma.q=0.1,
            x=x, sigma=sigma, mu0 = mu0, sigma0 = sigma0)

plot(theta, type="l")
abline(h=mu.n-1.96*sigma.n, col="green")
abline(h=mu.n+1.96*sigma.n, col="green")
abline(h=mu.n-3*sigma.n, col="red")
abline(h=mu.n+3*sigma.n, col="red")
```

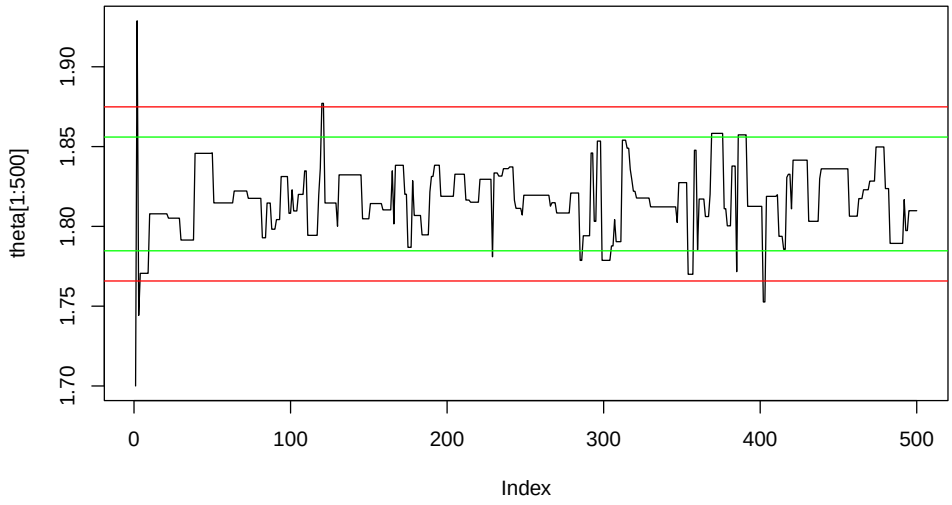


Na sliki smo za boljso predstavo oznacili 95% referencni interval prave aposterirone porazdelitve (zeleni crti) in odmik od povprecja aposteriorne porazdelitve za 3 standardne odklone, tj. 99.7% referencni interval (rdeci crti). **Pozor:** S tem smo uporabili vedenje o pravi aposteriorni porazdelitvi, ki v realni situaciji ni znana (ravno zato jo z MCMC metodami tudi ocenjujemo).

Bolj podrobno si ogledamo prvih 5000 iteracij:

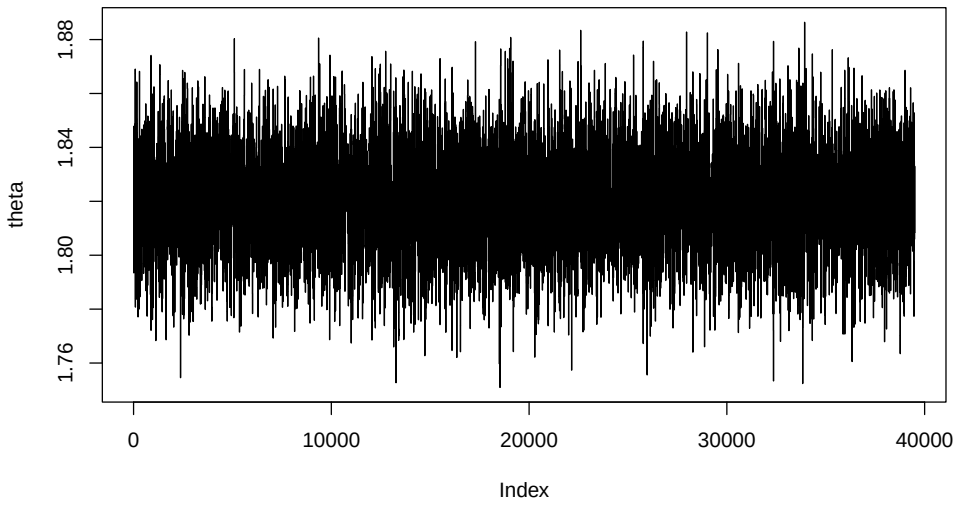


Bolj podrobno si ogledamo prvih 500 iteracij:



Potreben burn-in je zelo majhen, ker pa imamo veliko iteracij, vseeno vzamemo za burn-in kar 500:

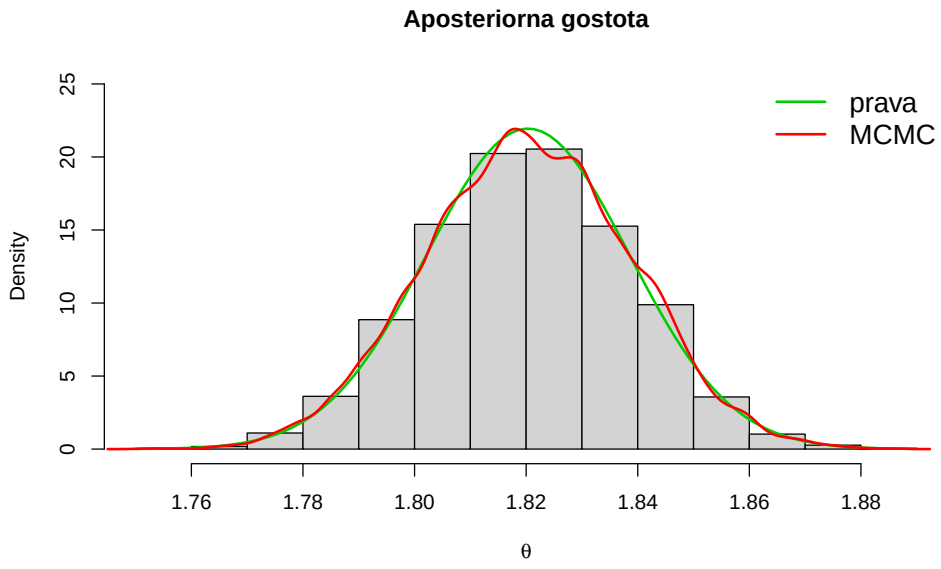
```
theta <- theta[-c(1:500)]
plot(theta, type="l")
```



Veriga torej izgleda primerno – skace gor in dol po nekem omejenem območju, "precesava" porazdelitev.

Primerjamo s pravo aposteriorno **porazdelitvijo – cilj konvergence je cela porazdelitev** (dobimo podobno):

```
### Narisemo
hist(theta, prob=T, main = "Aposteriorna gostota", xlab = expression(theta),
     ylim=c(0,25))
curve(dnorm(x, mean=mu.n, sd=sigma.n), add=T, col="green3", lwd=2)
lines(density(theta), col="red", lwd=2)
legend("topright",  lty = 1, lwd=2,
       c("prava", "MCMC"), col = c("green3","red"), bty = "n", cex = 1.3)
```



Primerjamo dobljeni tockovni oceni (dobimo podobno):

```
mean(theta) #MCMC
```

```
## [1] 1.820323
```

```
mu.n #teoreticno
```

```
## [1] 1.820331
```

Primerjamo dobljena kredibilna intervala (dobimo podobno):

```
library(HDInterval)
hdi(theta, credMass = 0.95) #MCMC
```

```
##    lower    upper
## 1.782952 1.853974
## attr(,"credMass")
## [1] 0.95
```

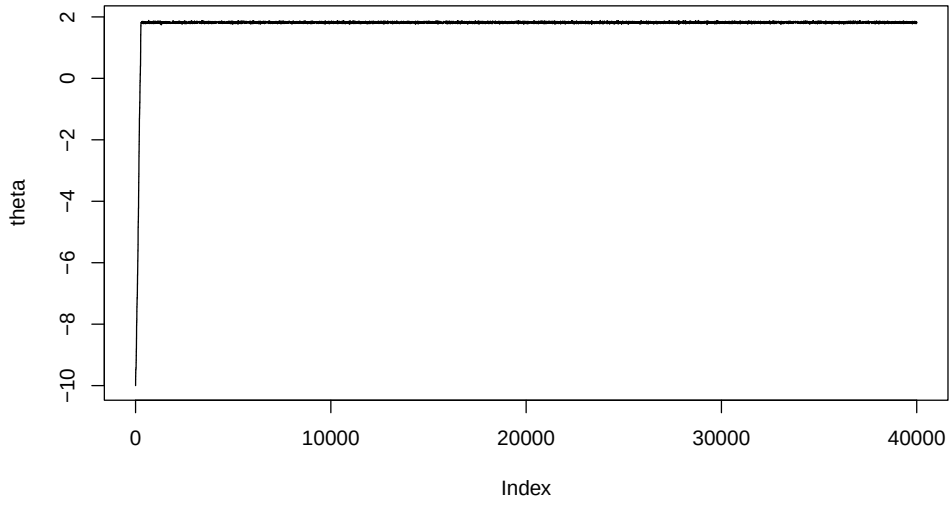
```
qnorm(c(0.025, 0.975), mean = mu.n, sd = sigma.n) #teoreticno
```

```
## [1] 1.784695 1.855966
```

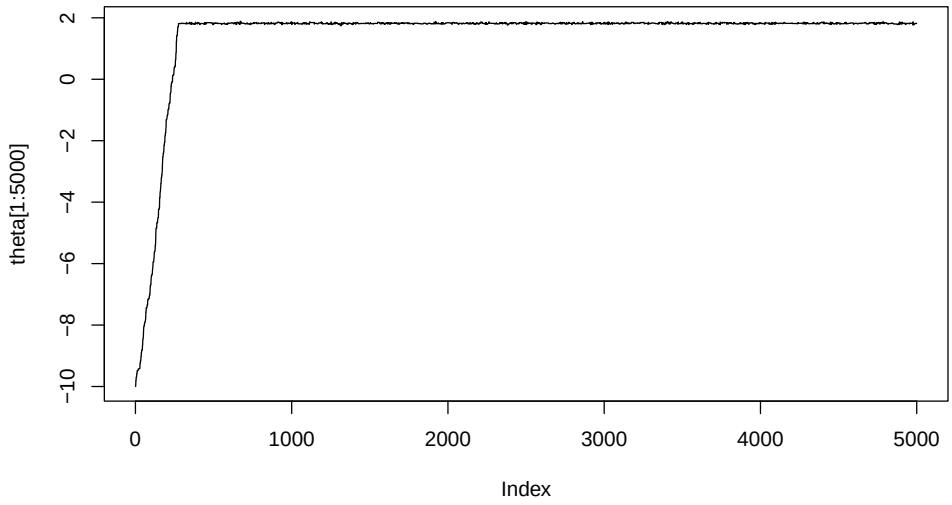
3.2.2 Primer z nesmiselno zacetno vrednostjo (varianca predlagalne porazdelitve se zmeraj "ustrezna")

Drasticno preizkusimo -10 kot zacetno vrednost – algoritem se zmeraj deluje, ker smo ga implementirali na logaritemski skali.

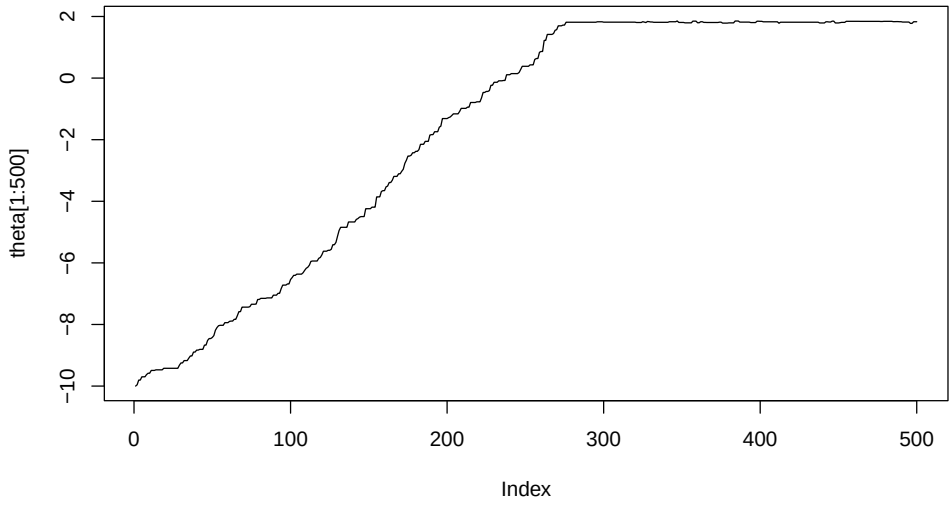
```
theta <- mh(n.iter=40000, theta.init=-10, sigma.q=0.1,
            x=x, sigma=sigma, mu0 = mu0, sigma0 = sigma0)
plot(theta, type="l")
```



Bolj podrobno si ogledamo prvih 5000 iteracij:

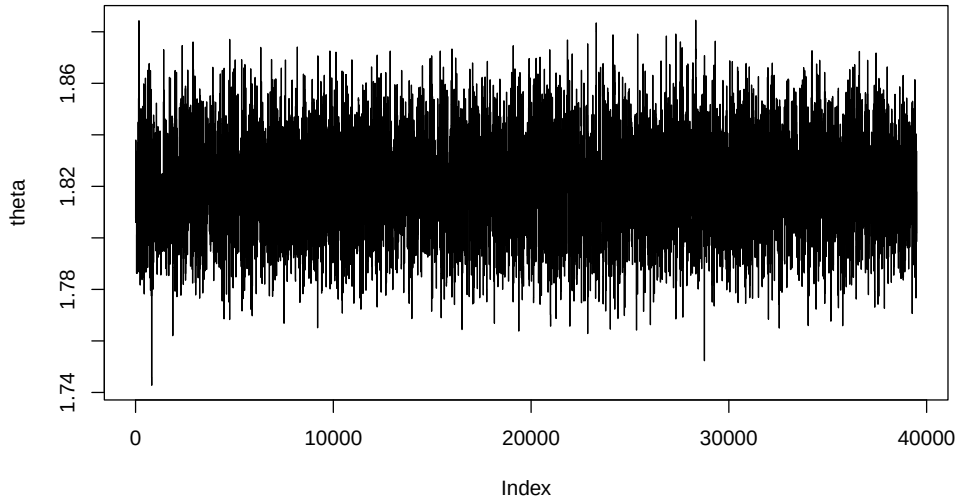


Bolj podrobno si ogledamo prvih 500 iteracij:



Odstranimo burn-in – je nujno, vendar kljub zelo nesmiselni zacetni vrednosti le-ta ni velik, saj imamo zelo preprost model z efektivnim vzorcevalnikom (*sampler*) :

```
theta <- theta[-c(1:500)]
plot(theta, type="l")
```



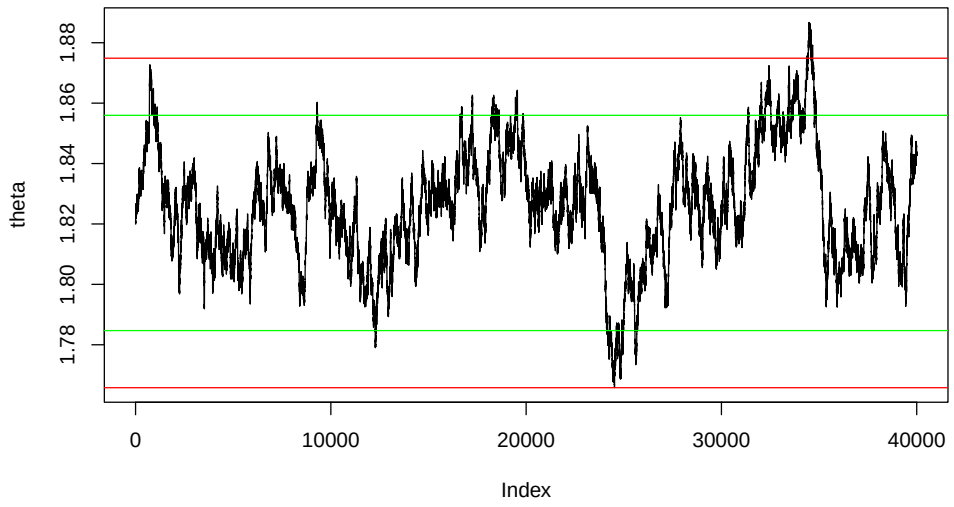
Taksna veriga izgleda primerno – **nas model in vzorcevalnik nista obcutljiva na izbor razlicnih zacetnih vrednosti** (pri bolj kompleksnih modelih seveda ne bi preizkusali tako cudnih zacetnih vrednostih, kot smo jih tu).

3.2.3 Primer z zelo majhno varianco predlagalne porazdelitve (pri tem je zacetna vrednost smiselna)

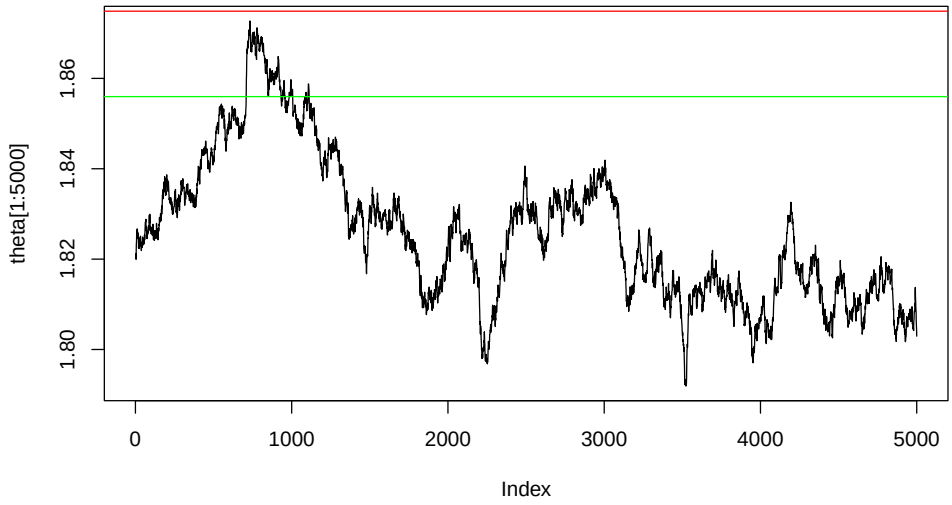
Za zacetno vrednost vzamemo povprecje vzorca, burn-in zato niti ne bomo potrebovali (v praksi vedno vzamemo burn-in, saj v kompleksnejsih modelih ponavadi zacnemo pri vsaj nekoliko prevec oddaljenih vrednostih). Za SO predlagalne porazdelitve vzamemo 0.001. V naslednjem koraku se bomo torej lahko oddaljili najvec do nekje $\pm 3 \cdot 0.001 = \pm 0.003$, tj. 3 mm — omejimo se na zelo majhno obmocje okoli prejsnje $\theta^{(i)}$. Ko smo torej na nekem obmocju aposteriorne porazdelitve, bomo lahko presli na drugi konec porazdelitve v zelo kratkih korakih in bomo zato aposterirono porazdelitev raziskovali zelo pocasi.

Narisemo dobljeni vzorec:

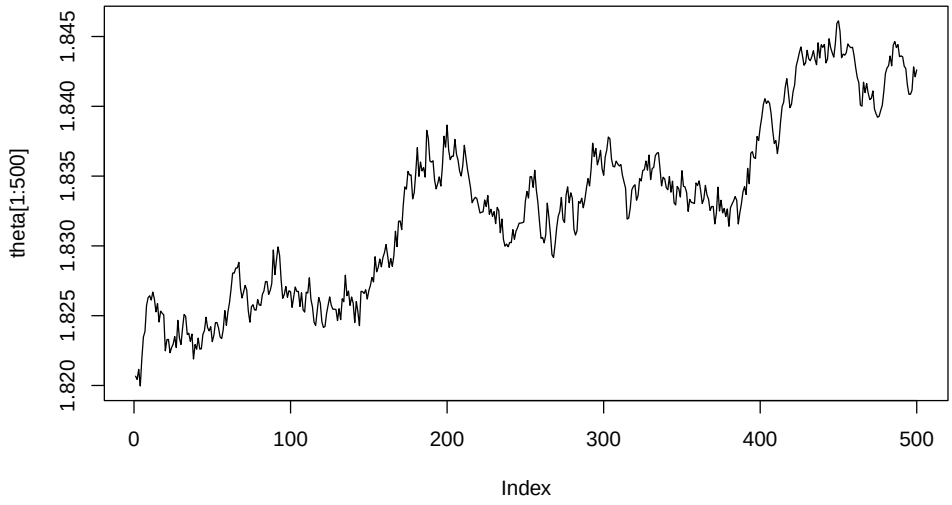
```
theta <- mh(n.iter=40000, theta.init=mean(x), sigma.q=0.001,
            x=x, sigma=sigma, mu0 = mu0, sigma0 = sigma0)

plot(theta, type="l")
abline(h=mu.n-1.96*sigma.n, col="green")
abline(h=mu.n+1.96*sigma.n, col="green")
abline(h=mu.n-3*sigma.n, col="red")
abline(h=mu.n+3*sigma.n, col="red")
```



Bolj podrobno si ogledamo prvih 5000 iteracij:

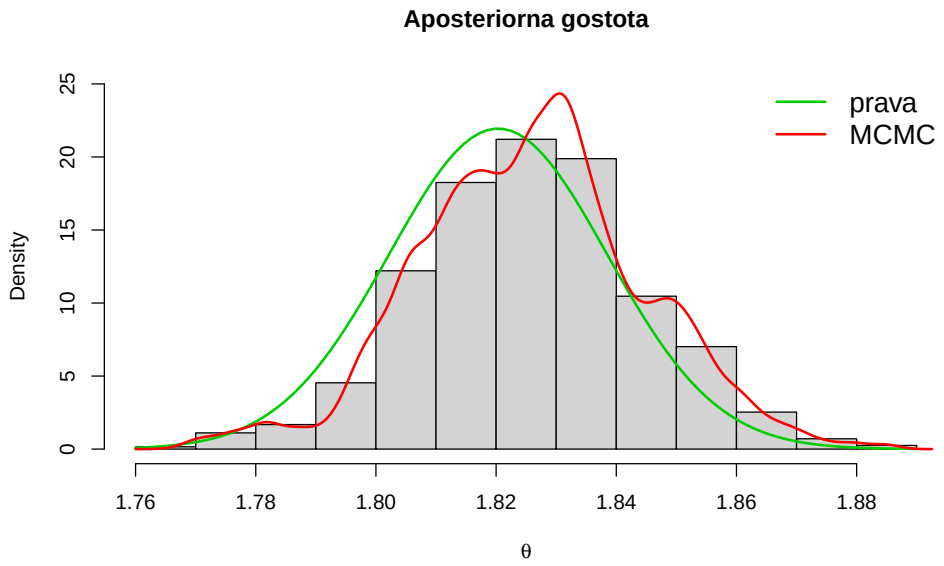


Bolj podrobno si ogledamo prvih 500 iteracij:

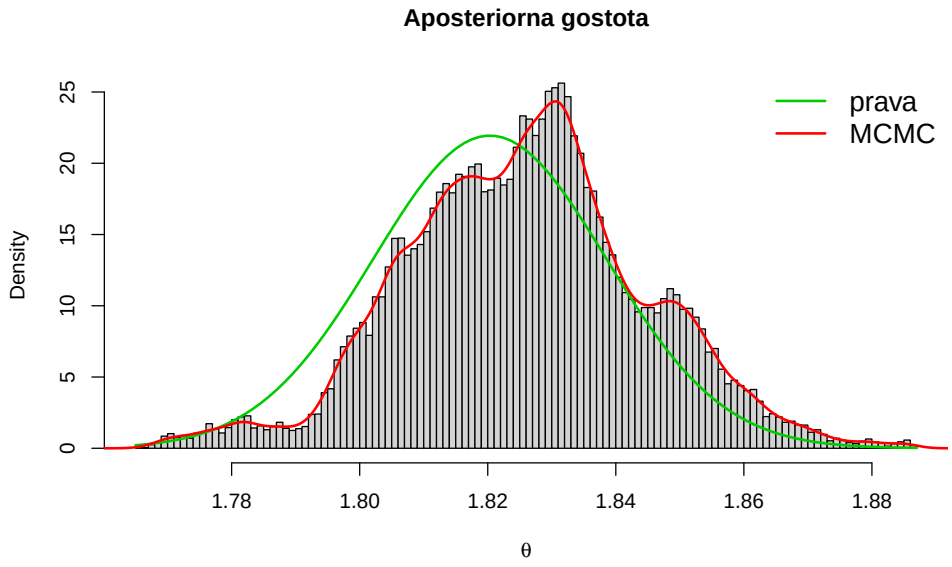


Nase razmisljanje o problemih ob premajhni varianci predlagalne porazdelitve se je torej potrdilo – **porazdelitev prepocasi raziskujemo, tj. iteracije so med seboj prevec korelirane** (vec o tem spodaj).

Najbolj pomembno je seveda, kaj dobimo za rezultat, tj. aposteriorno porazdelitev:



Za boljsi prikaz povecamo stevilo stolpcev, tako da je bolj jasna oblika zglajene gostote (rdeca krivulja):



Nas priblizek za aposteriorno porazdelitev je torej obcutno slabsi kot pri prvem primeru (`sigma.q = 0.001`) ob enakem stevilu iteracij, tj. ob enaki casovni in prostorski zahtevnosti algoritma.

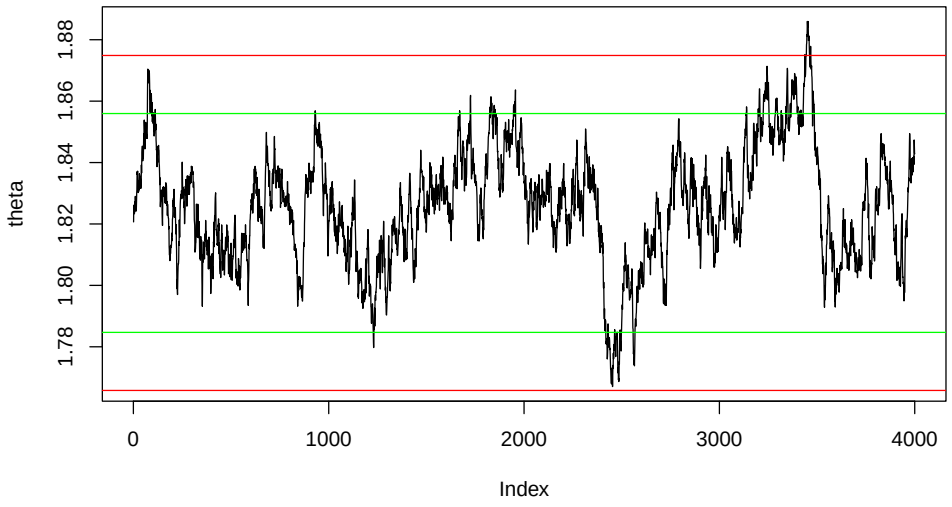
Ali *thinning* pomaga?

O tem bomo sicer podrobneje govorili na naslednjih vajah. Na kratko: v koncni vzorec vzamemo vsakega k-tega. Ideja je, da se s tem izognemo korelaciji med cleni verige, tj. avtokorelaciji. Le-ta je pri MCMC metodah vedno prisotna zaradi same narave vzorcenja, avtokoreliran vzorec pa ne moremo imeti za slucajni vzorec (neodvisne enako porazdeljene, IID) iz porazdelitve.

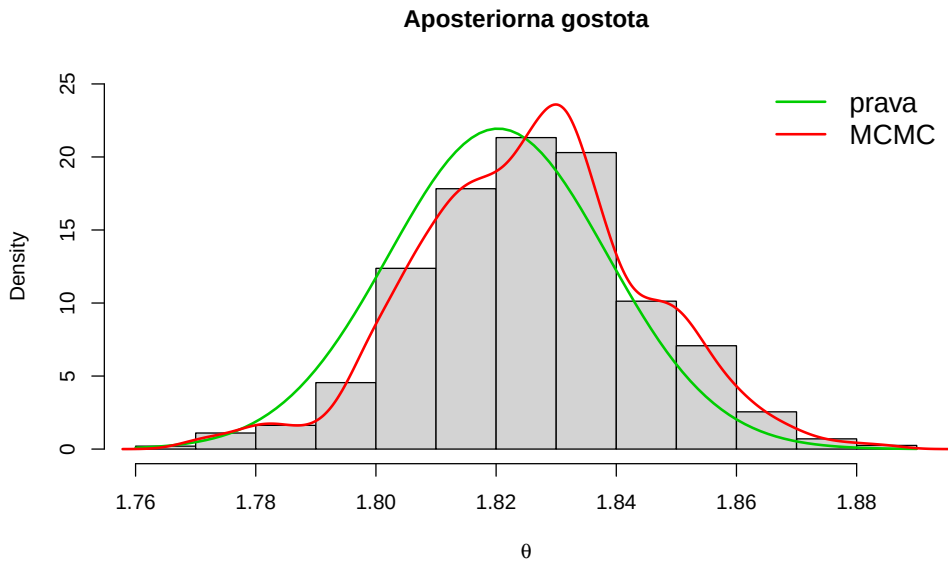
V vzorec vzamemo vsakega 10-tega (velik faktor):

```
theta <- theta[seq(1, length(theta), by = 10)]
```

Razredcena veriga – malo bolje, vendar se zmeraj kar mocna avtokorelacija, saj smo vzeli res zelo majhen SO predlagalne porazdelitve:



Dobljena aposteriorna porazdelitev:



Zavedati se moramo predvsem, da smo dobesedno vrgli stran 90 % ze izracunanih clenov, casovna zahtevnost torej ostaja enaka ob bistveno manjsem koncnem vzorcu. Prihranili bi lahko le pri prostorski zahtevnosti, saj bi lahko algoritem implementirali tako, da bi si po desetih izracunanih iteracijah shranili le zadnjega, preostale pa pozabili. Ob casovni zahtevnosti na eni strani in velikosti vzorca iz aposteriorne porazdelitve na drugi strani (pa ceprav je avtokoreliran), je prostorska zahtevnost pravzaprav edina prednost *thinninga*, kar pa v dobi modernih raucnalnikov ni vec taksen problem. **Redcenje verig (*thinning*) se torej v praksi opusca, porocamo pa t.i. *effective sample size***, tj. stevilo neodvisnih opazovanj, ki se izracuna tako, da se ob stevilu iteracij upposteva avtokorelacije vzorca – vec o tem v 6. sklopu.

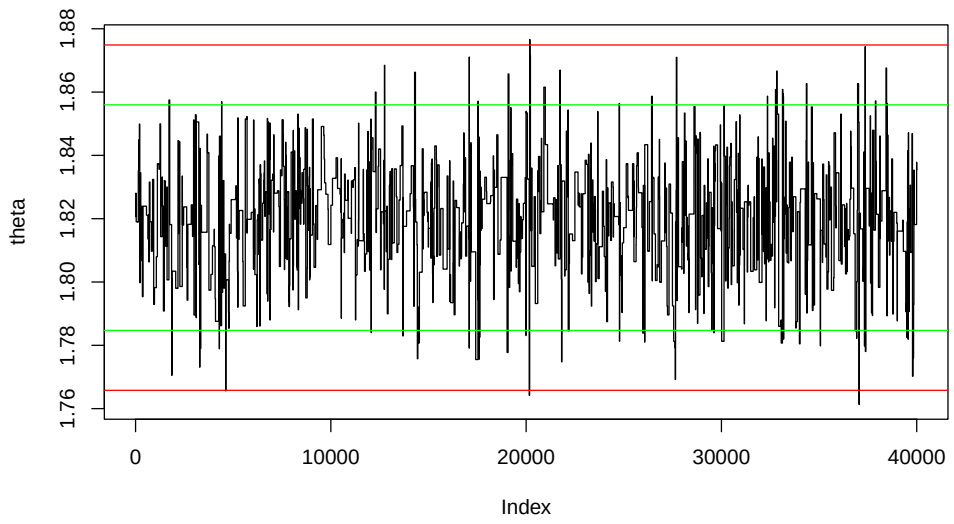
Seveda pa je prvi korak ob taksni verigi, da vzamemo bolj primeren vzorcevalnik oz. razmislimo o smiselnosti modela, ce se tezave zopet pojavijo.

3.2.4 Primer z zelo veliko varianco predlagalne porazdelitve (pri tem je zacetna vrednost smiselna)

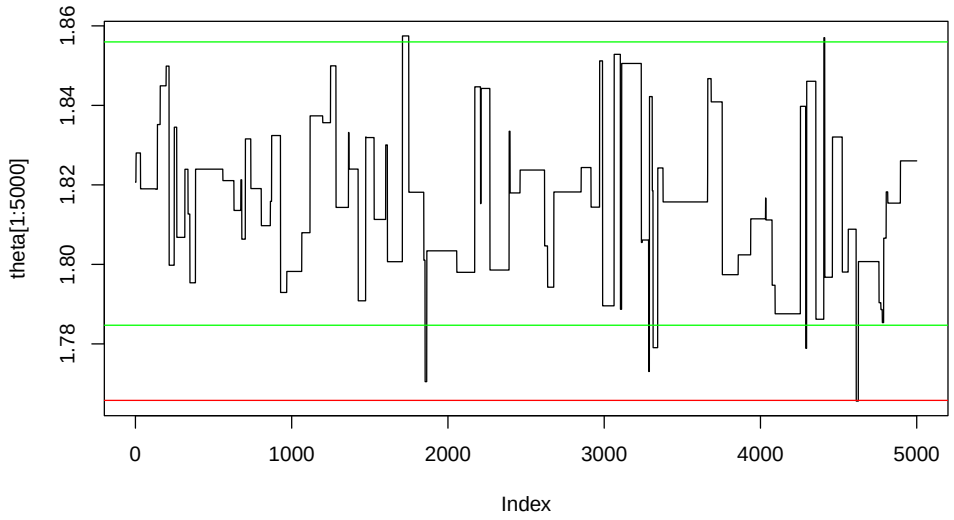
Spet vzamemo za zacetno vrednost povprecje vzorca, za SO predlagalne porazdelitve pa 1. V naslednjem koraku se bomo torej lahko oddaljili najvec do nekje $\pm 3 \cdot 1 = \pm 3$ metre – kandidati za novo vrednost θ^* lahko torej nekontrolirano skacejo okoli prejsnje $\theta^{(i)}$, zaradi cesar so z zelo veliko verjetnostjo neprimerni, manj primerni od prejsnjega. Ko smo torej na neki dokaj smiselni vrednosti iz aposteriorne porazdelitve, se bomo z zelo majhno verjetnostjo premaknili nekam drugam – **tratimo cas, medtem ko ne dobivamo novih opazovanj iz aposteriorne porazdelitve**.

Narisemo dobljeni vzorec:

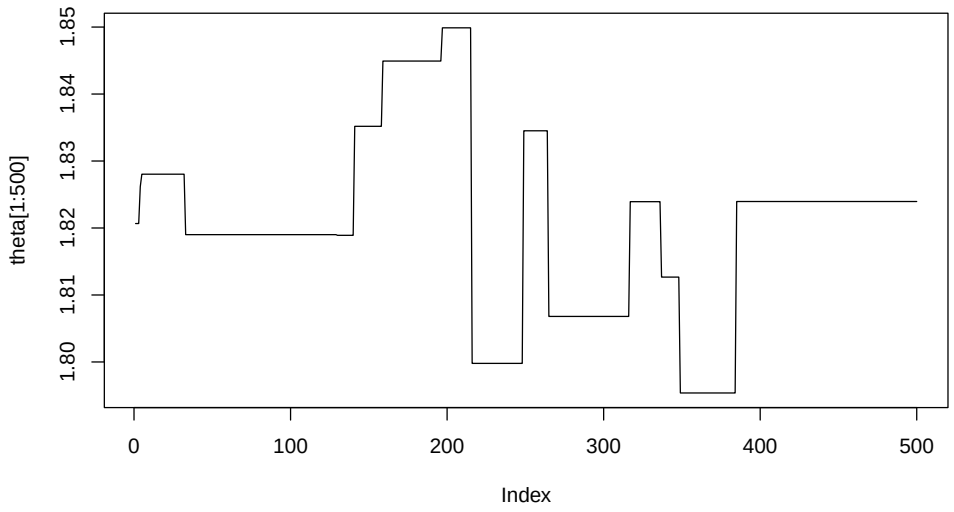
```
theta <- mh(n.iter=40000, theta.init=mean(x), sigma.q=1,
            x=x, sigma=sigma, mu0 = mu0, sigma0 = sigma0)

plot(theta, type="l")
abline(h=mu.n-1.96*sigma.n, col="green")
abline(h=mu.n+1.96*sigma.n, col="green")
abline(h=mu.n-3*sigma.n, col="red")
abline(h=mu.n+3*sigma.n, col="red")
```



Bolj podrobno si ogledamo prvih 5000 iteracij:

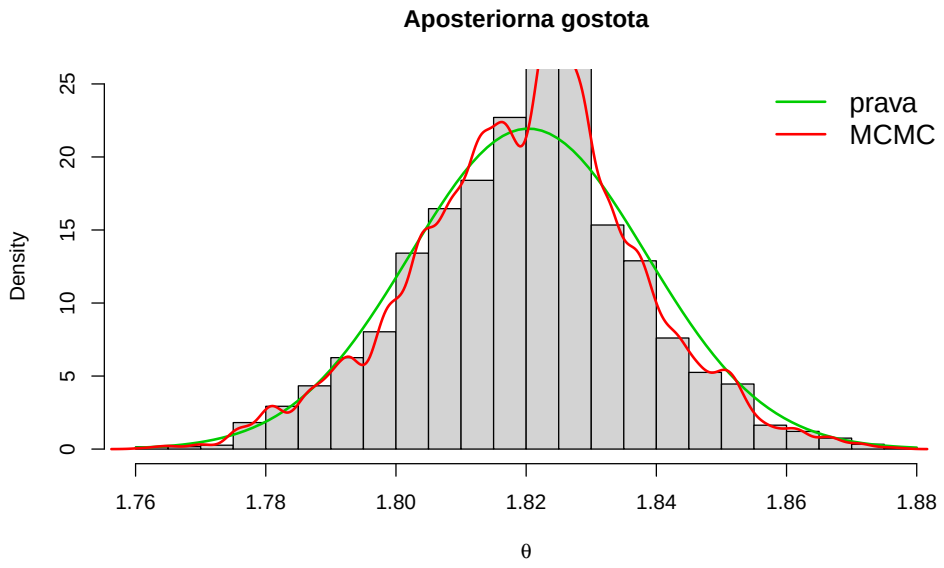


Bolj podrobno si ogledamo prvih 500 iteracij:

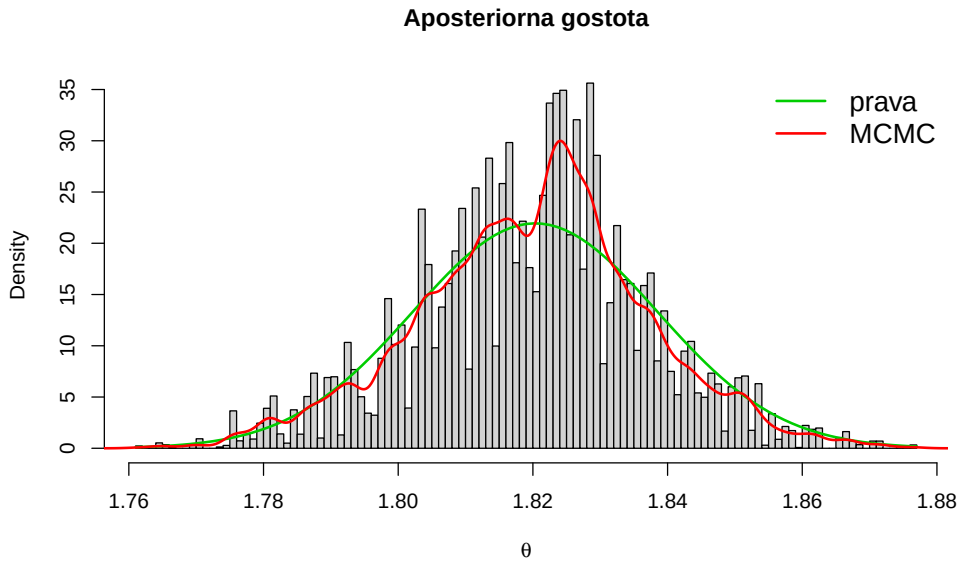


Nase razmisljanje o problemih ob preveliki varianci predlagalne porazdelitve se je torej potrdilo – veriga je velikokrat konstantna za veliko stevilo iteracij.

Dobljena aposteriorna porazdelitev:



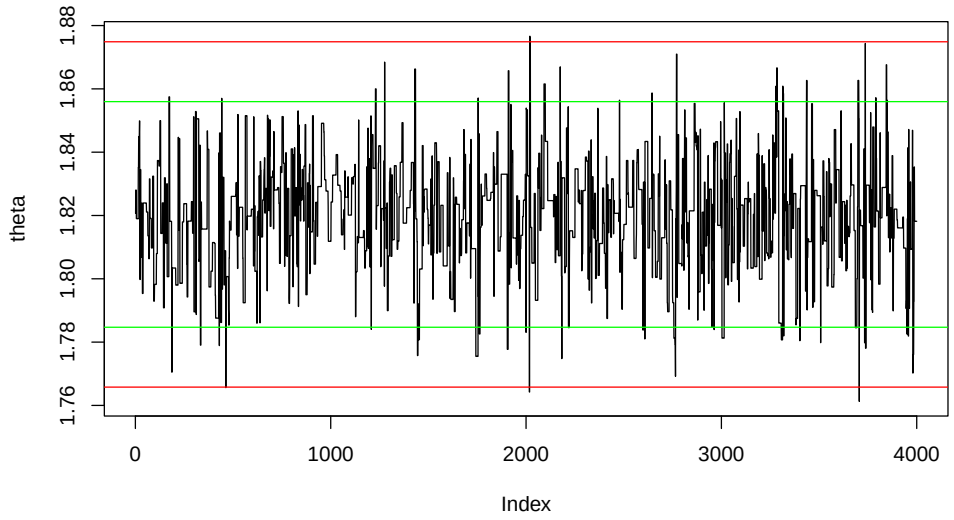
Za boljsi prikaz povecamo stevilo stolpcev – tako so postanejo se bolj jasni vrhovi zglajene gostote (rdeca krivulja), ki se primerijo tam, kjer je veriga konstantna za veliko stevilo iteracij:

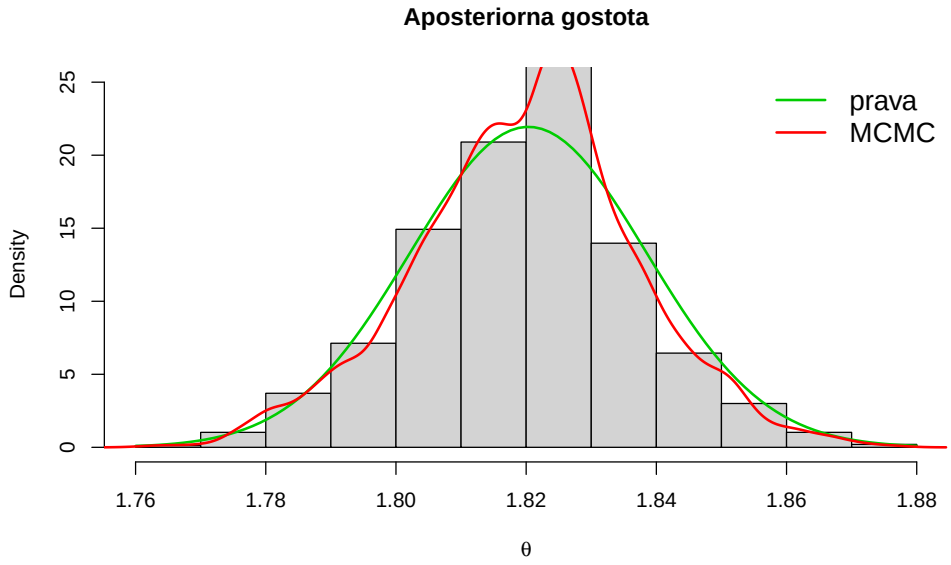


Nas priblizek za aposteriorno porazdelitev je torej tudi tu obcutno slabsi kot pri prvem primeru (`sigma.q = 0.001`) ob enaki casovni in prostorski zahtevnosti algoritma.

Ali *thinning* pomaga?

```
theta <- theta[seq(1, length(theta), by = 10)]
```





Podoben komentar kot prej – navidezno pomaga, saj je veriga bolj primerna, vendar imamo majhen vzorec ob veliki casovni zahtevnosti.

3.2.5 Kako izbrati primeno varianco predlagalne porzdelitve?

Algoritmi vzorcenja, ki so implementirani v knjiznicah, morajo seveda dobro delovati na sirokem naboru modelov oz. spremenljivk, predvsem bi bilo absurdno, da algoritem ne bi deloval ob spremembi enot spremenljivke – primerna velikost variance predlagalne porazdelitve je namrec odvisna od samih vrednosti spremenljivke.

Problem pri zgornjih dveh primerih je bil:

- Pri premajhni varianci je algoritem sprejel za novo vrednost skoraj vsakega kandidata θ^*, tj. delez sprejetih (*acceptance rate*) je bil prevelik.
- Pri preveliki varianci je algoritem malokrat sprejel kandidata θ^* za novo vrednost, tj. delez sprejetih je bil premajhen.

Resitev je *adaptive* algoritem, kjer vsakih npr. 200 korakov izracunamo, koliksen delez kandidatov smo v zadnjih 200 iteracijah sprejeli. Ce je delez premajhen ali prevelik, potem varianco predlagalne porazdelitve zmanjsamo oz. povecamo.

Kaj pomeni premajhen/prevelik delez sprejetih oz. kaksen delez sprejetih je optimalen? Tipicno je to 0.234 (teoreticno dokazano v nekih okvirih; vseeno ni vedno res najbolje). Fascinantno natancna stevilka za splosno uporabo, ki sicer presenetljivo ni 0.42 :)

4 Metropolis-Hastings za dvoparametricni model

Za primer vzamemo normalni model z dvema parametroma $\theta = (\mu, \tau)$, kjer je $\tau = 1/\sigma^2$, za nas vzorec x (glejte 5. sklop), kjer za apriorno porazdelitev vzamemo

$$\pi(\mu, \tau) = \pi(\mu) \cdot \pi(\tau), \quad \mu \sim N(0, \tau_0 = 0.001), \quad \tau \sim \text{Gama}(0.01, 0.01).$$

Vzeli smo sibko informativno porazdelitev (neodvisnost parametrov kot pri Jeffreyevi apriorni, vendar vzamemo pravi porazdelitvi za vsak parameter).

Za predlagalno porazdelitev si moramo izbrati dvorazsezno porazdelitev. Nov predlog za θ bomo vzorcili tako, da bomo neodvisno vzorcili nova predloga za μ in τ, sprejeli ali zavrnili pa ju bomo skupaj.

Nov predlog za μ bomo vzorcili iz $N(\mu^{(i-1)}, \tau_{\text{predlog}} = 10)$, kjer je τ_{predlog} *precision* predlagalne porazdelitve, τ pa iz log-normalne z ustreznima parametroma.

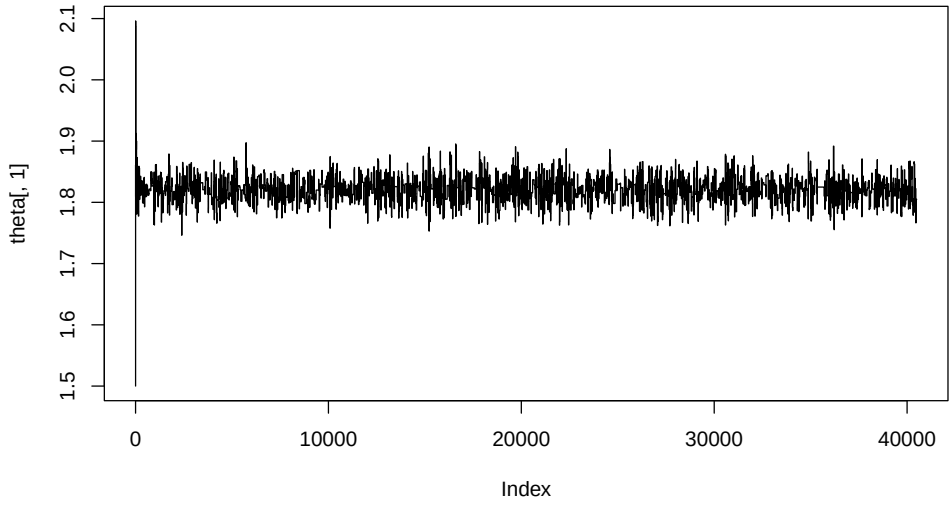
```
#Logaritem verjetja
loglik <- function(x, theta) {
   sum(dnorm(x, mean = theta[1], sd = sqrt(1 / theta[2]), log = TRUE))
}

#Apriorna porazdelitev (log)
logprior <- function(theta) {
  #na log skali sestejemo gostoti zaradi neodvisnosti
  dnorm(theta[1], 0, sd = sqrt(1 / 0.001), log = TRUE) +
    dgamma(theta[2], 0.01, 0.01, log = TRUE)
}

#Predlagalna porazdelitev - vzorcenje
rq <- function(theta) {
  rez <- c(NA, NA)
  #neodvisno vzorcimo vsak parameter posebej
  rez[1] <- rnorm(1, theta[1], sd = sqrt(1 / 10))
  rez[2] <- rlnorm(1, meanlog = log(theta[2]), sdlog = sqrt(1 / 10))
  return(rez)
}

#Predlagalna porazdelitev - gostota (log)
logdq <- function(theta.arg, theta.cond) {
  #na log skali sestejemo gostoti zaradi neodvisnosti
  dnorm(theta.arg[1], theta.cond[1], sd = sqrt(1 / 10), log = TRUE) +
    dlnorm(theta.arg[2], meanlog = log(theta.cond[2]), sdlog = sqrt(1 / 10), log = TRUE)
}
```

```r
#Enak algoritem kot prej,
#le da prilagodimo za racunanje in shranjevanje dveh parametrov.
#Ker uporabimo seznam (list), deluje spodnje za poljubno stevilo parametrov
#(doloceno z uporabljenimi funkcijami zgoraj).
mh2 <- function(n.iter=40500, theta.init=c(1.5, 1), x=x) {
  theta <- as.list(rep(NA, n.iter))
  theta[[1]] <- theta.init

  for(i in 2:n.iter) {
    new.theta <- rq(theta[[i - 1]])

    #Log-Accepance probability
    logacc.prob <- loglik(x, new.theta) +
      logprior(new.theta) +
      logdq(theta[[i - 1]], new.theta) -
      loglik(x, theta[[i - 1]]) -
      logprior(theta[[i - 1]]) -
      logdq(new.theta, theta[[i - 1]])
    logacc.prob <- min(0, logacc.prob)#0 = log(1)

    if(log(runif(1)) < logacc.prob) {
      #Accept
      theta[[i]] <- new.theta
    } else {
      #Reject
      theta[[i]] <- theta[[i - 1]]
    }
  }
  return(do.call(rbind, theta))
}
```

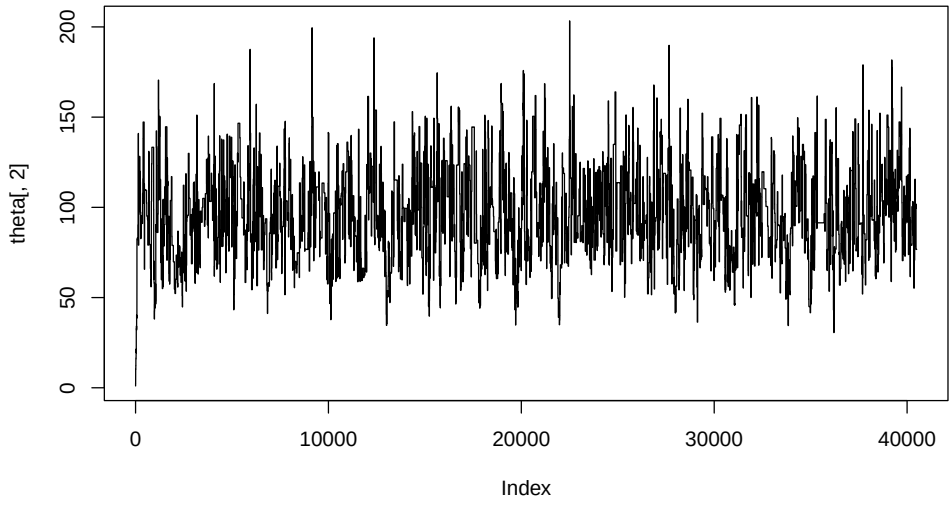
Pozenemo algoritem:

```r
theta <- mh2(n.iter=40500, theta.init=c(1.5, 1), x=x)
```

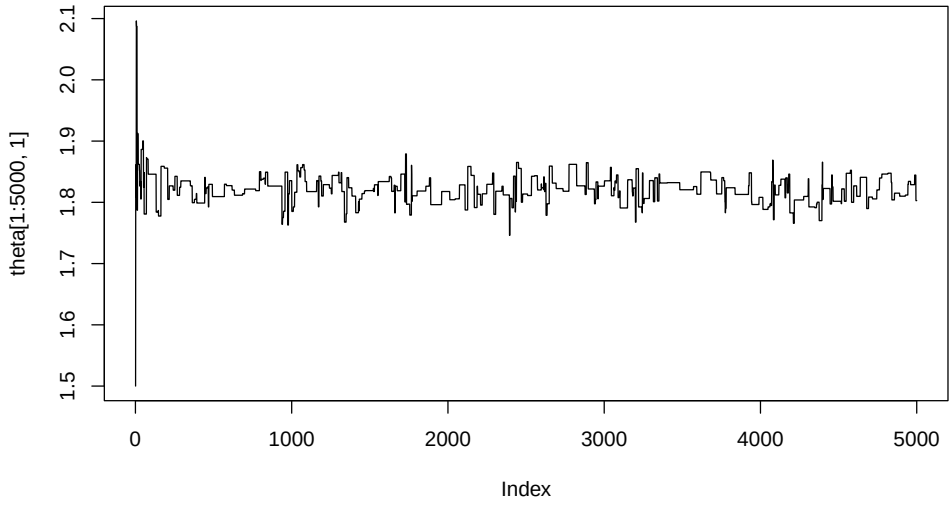
Veriga za μ:



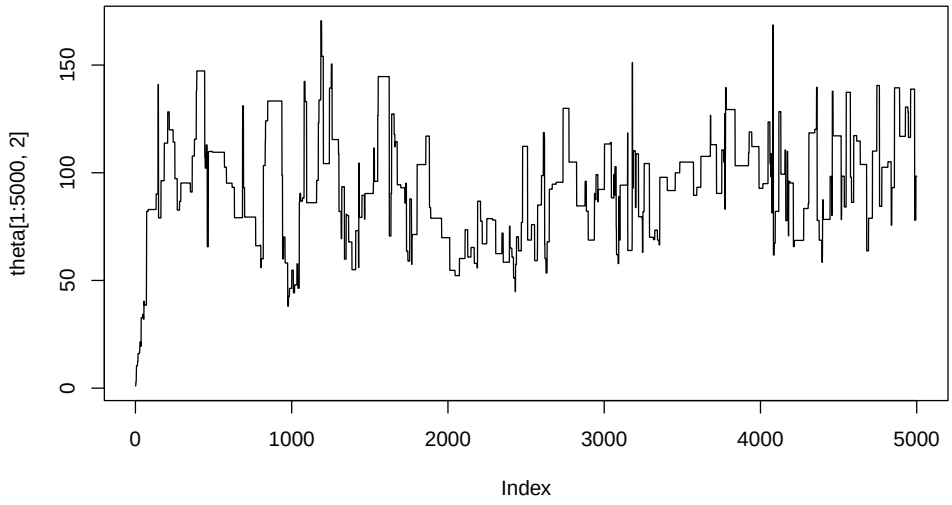
Veriga za τ:



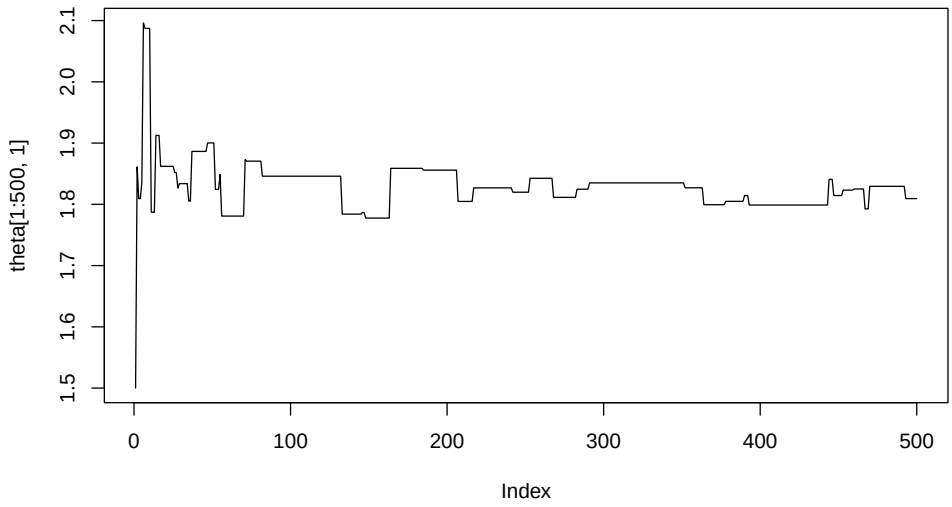
Veriga za μ – prvih 5000 iteracij:



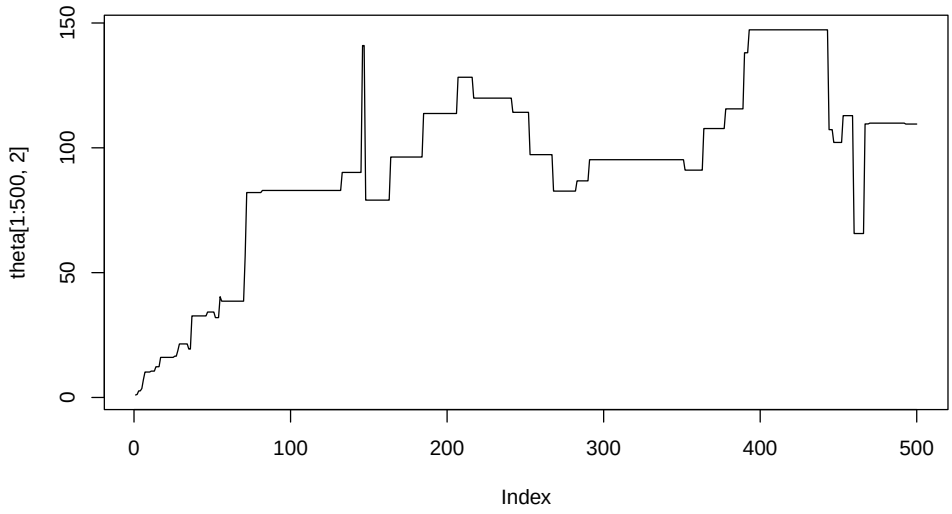
Veriga za τ – prvih 5000 iteracij:



Veriga za μ – prvih 500 iteracij:



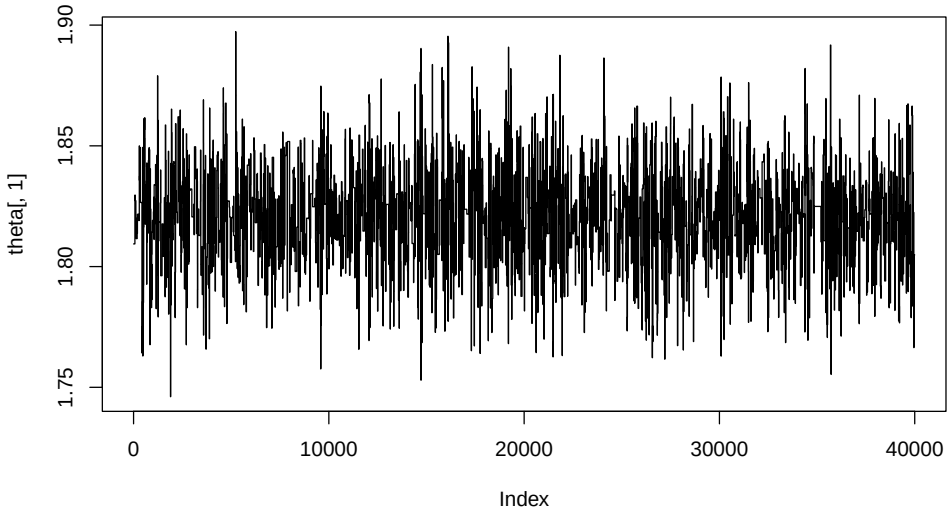
Veriga za τ – prvih 500 iteracij:



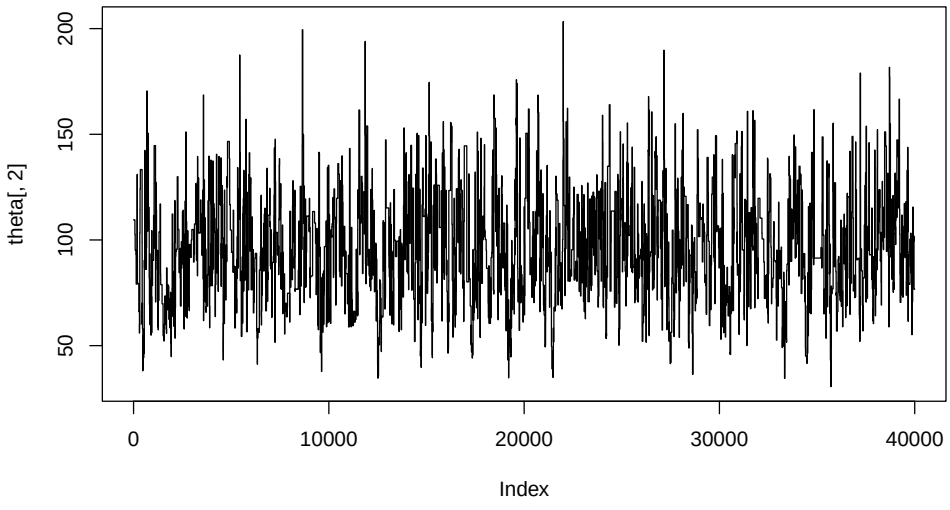
Odstranimo zacetnih nekaj clenov (*burn-in*):

```
theta <- theta[-c(1:500),]
```

Koncna veriga za μ:



Koncna veriga za τ:

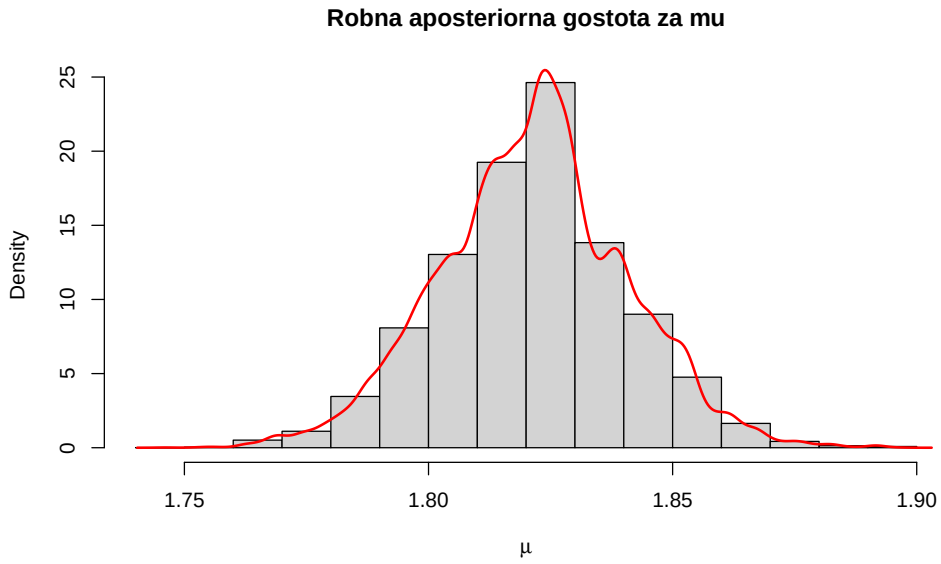


Dobljene robne aposteriorne porazdelitve:

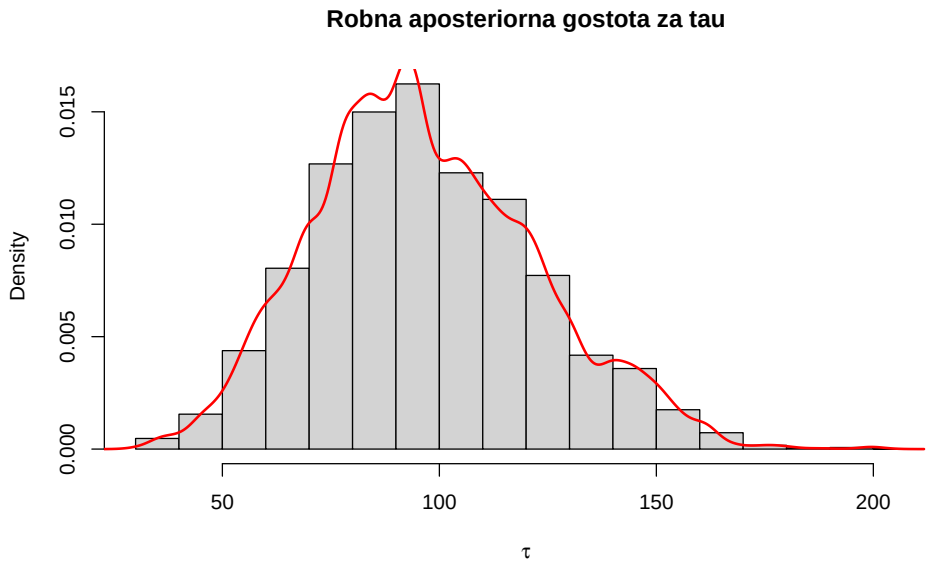
```
apply(theta, 2, summary)
```

```
##              [,1]      [,2]
## Min.    1.746096  30.59755
## 1st Qu. 1.809384  78.69936
## Median  1.821900  94.16733
## Mean    1.821376  96.74281
## 3rd Qu. 1.833142 113.66268
## Max.    1.897310 203.37478
```

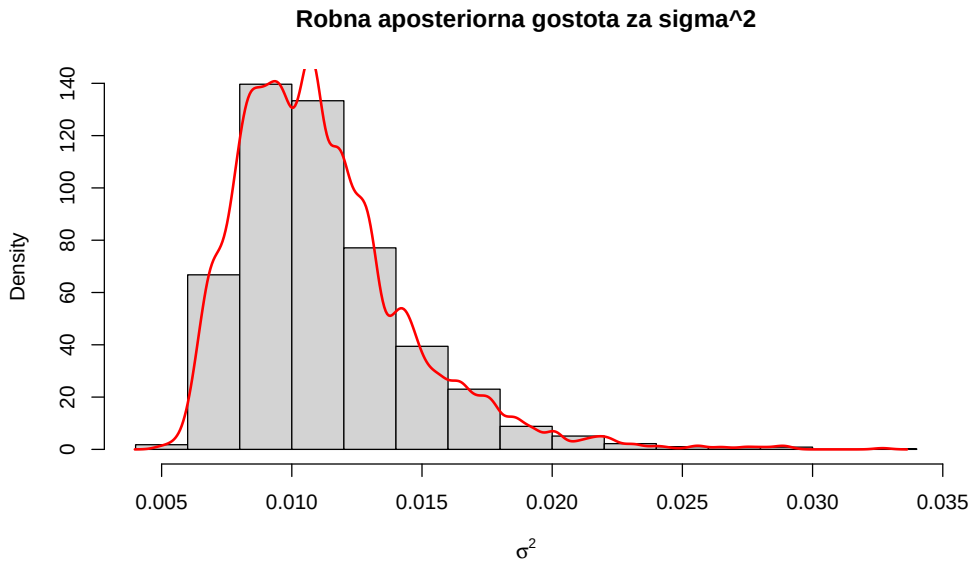
```
hist(theta[,1], prob=T, main = "Robna aposteriorna gostota za mu",
     xlab = expression(mu))
lines(density(theta[,1]), col="red", lwd=2)
```




```
hist(theta[,2], prob=T, main = "Robna aposteriorna gostota za tau",
     xlab = expression(tau))
lines(density(theta[,2]), col="red", lwd=2)
```




```
hist(1/theta[,2], prob=T, main = "Robna aposteriorna gostota za sigma^2",
     xlab = expression(sigma^2))
lines(density(1/theta[,2]), col="red", lwd=2)
```



Vzorec iz robne aposteriorne porazdelitve za σ^2 torej dobimo iz vzorcev za τ na preprost nacin (zadnji graf).

5. sklop: Normalni model z dvema parametroma

1 Primer

Podan imamo naslednji vzorec visin (metri) studentov moskega spola:

```
x <- c(1.91, 1.94, 1.68, 1.75, 1.81, 1.83, 1.91, 1.95, 1.77, 1.98,
       1.81, 1.75, 1.89, 1.89, 1.83, 1.89, 1.99, 1.65, 1.82, 1.65,
       1.73, 1.73, 1.88, 1.81, 1.84, 1.83, 1.84, 1.72, 1.91, 1.63)
```

Zanima nas povprecna visina studentov, kjer standardnega odklona ne poznamo.

2 Verjetnostni model za nas primer

Vzorec $X_1, X_2, \ldots, X_n$, kjer je:

- $n = 30$ stevilo studentov,
- X_i predstavlja visino i-tega studenta,
- $X_i \mid \mu, \sigma^2 \sim N(\mu, \sigma^2)$,
- $f(x \mid \mu, \sigma^2) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$.

Prostor parametrov je dvorazsezen: $\theta = (\mu, \sigma^2) \in \mathbb{R} \times (0, \infty)$.

3 Ocenjevanje v Bayesovi statistiki

Bayesova formula:

$$\pi(\mu, \sigma^2 \mid x) \propto L(\mu, \sigma^2 \mid x)\, \pi(\mu, \sigma^2).$$

Nastaviti moramo torej dvorazsezno apriorno porazdelitev in dobiti dvorazsezno aposteriorno porazdelitev.

3.1 Verjetje

$$L(\mu, \sigma^2 \mid x) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x_i-\mu)^2}{2\sigma^2}}$$

3.2 Apriorna porazdelitev

Vedno velja (pogojna verjetnost):

$$\pi(\mu, \sigma^2) = \pi(\mu \mid \sigma^2) \cdot \pi(\sigma^2).$$

3.2.1 Konjugirana apriorna porazdelitev

Za $\mu \mid \sigma^2$ vzamemo

$$\mu \mid \sigma^2 \sim N(\mu_0, \sigma^2/\kappa_0).$$

Porazdelitev σ^2 lahko zapisemo na tri ekvivalentne nacine:

$$\sigma^2 \sim \text{Inv-Gama}(\nu_0/2,\ \nu_0\,\sigma_0^2/2),$$

kar pomeni ravno

$$1/\sigma^2 \sim \text{Gama}(\nu_0/2,\ \nu_0\,\sigma_0^2/2),$$

kar je ekvivalentno

$$\nu_0\,\sigma_0^2/\sigma^2 \sim \text{Gama}(\nu_0/2,\ 1/2) = \chi^2_{\nu_0}.$$

Ideja za parametre apriorne porazdelitve:

- μ_0 je nase prepricanje o μ
- σ_0^2 je nase prepricanje o σ^2 (ker je ravno pricakovana vrednost)
- vecji kot je ν_0 oz. κ_0, bolj verjamete, da je μ enak μ_0 oz. σ^2 enak σ_0^2
- κ_0 oz. ν_0 predstavljata velikost vzorca, na podlagi katerega je vase prepricanje o μ oz. σ^2 osnovano

3.2.2 Neinformativna apriorna porazdelitev

Privzamemo neodvisnost apriornih porazdelitev parametrov:

$$\pi(\mu, \sigma^2) = \pi(\mu) \cdot \pi(\sigma^2).$$

Jeffreyeva apriorna porazdelitev za μ v normalnem modelu z znanim σ^2:

$$\pi(\mu) \propto \sqrt{1/\sigma^2} \propto 1.$$

Jeffreyeva apriorna porazdelitev za σ^2 v normalnem modelu z znanim μ:

$$\pi(\sigma^2) \propto 1/\sigma^2.$$

Dobimo torej

$$\pi(\mu, \sigma^2) \propto 1/\sigma^2,$$

kar je seveda *improper* porazdelitev.

3.3 Aposteriorna porazdelitev

Tudi tu vedno velja (pogojna verjetnost):

$$\pi(\mu, \sigma^2 \mid x) = \pi(\mu \mid \sigma^2, x) \cdot \pi(\sigma^2 \mid x).$$

3.3.1 Aposteriorna porazdelitev pri konjugirani apriorni porazdelitvi

Dobimo:

$$\mu \mid \sigma^2, x \sim N(\mu_n, \sigma^2/\kappa_n),$$

kjer je

$$\kappa_n = \kappa_0 + n,$$
$$\mu_n = \frac{\kappa_0}{\kappa_0 + n}\mu_0 + \frac{n}{\kappa_0 + n}\bar{x}.$$

Aposteriorno porazdelitev σ^2 lahko zapisemo na tri ekvivalentne nacine:

$$\sigma^2 \mid x \sim \text{Inv-Gama}(\nu_n/2,\ \nu_n\,\sigma_n^2/2) \iff 1/\sigma^2 \mid x \sim \text{Gama}(\nu_n/2,\ \nu_n\,\sigma_n^2/2) \iff$$
$$\iff \nu_n\,\sigma_n^2/\sigma^2 \mid x \sim \text{Gama}(\nu_n/2,\ 1/2) = \chi^2_{\nu_n},$$

kjer je

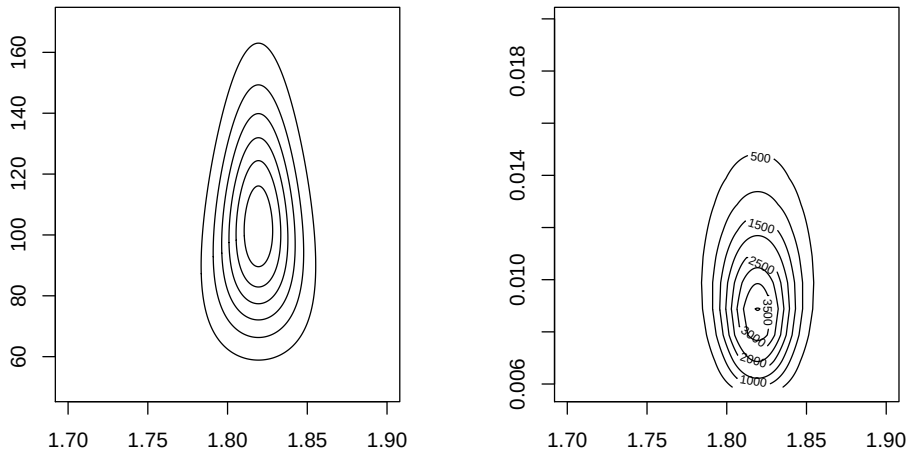
$$\nu_n = \nu_0 + n,$$
$$\nu_n\,\sigma_n^2 = \nu_0\,\sigma_0^2 + (n-1)s^2 + \frac{n\kappa_0}{\kappa_0 + n}(\bar{x} - \mu_0)^2.$$

Ideja:

- Aposteriorna pricakovana vrednost μ_n je utezeno povprecje apriorne pricakovane vrednosti μ_0 in vzorcnega povprecja $\bar{x}$, kjer preko κ_0 kontroliramo, kako mocno verjamemo apriorni pricakovani vrednosti, medtem ko n doloca, kako mocno verjamemo vzorcu.
- Na ravni vsote kvadratov odstopanj je aposteriorna enaka vsoti apriorne, vzorcne in zadnjega clena, ki je posledica pogojne odvisnosti μ od σ^2.

```
### Izberemo parametre apriorne porazdelitve
mu.0 <- 1.78
kappa.0 <- 1

sigma2.0 <- 0.1^2
nu.0 <- 1

### Izracunamo parametre aposteriorne porazdelitve
n <- length(x)

kappa.n <- kappa.0 + n
mu.n <- kappa.0/kappa.n * mu.0 + n/kappa.n * mean(x)

nu.n <- nu.0 + n
```

```r
sigma2.n <- ( nu.0*sigma2.0 + (n-1)*var(x) +
                n*kappa.0/kappa.n * (mean(x)-mu.0)^2 ) / nu.n

### Narisemo porazdelitev
# Prvi nacin: s parametrom 1/sigma^2
mu <- seq(1.7, 1.9, 0.0001)
prec2 <- seq(50, 170, 1)

apost <- matrix(NA, nrow = length(mu), ncol = length(prec2))
for (i in 1:length(mu)) {
  for (j in 1:length(prec2)) {
    apost[i, j] = dnorm(mu[i], mean = mu.n, sd = sqrt(1/(kappa.n*prec2[j]))) *
      dgamma(prec2[j], nu.n/2, nu.n*sigma2.n/2)
  }
}

par(mfrow = c(1,2))
contour(mu, prec2, apost)

# Drugi nacin: s parametrom sigma^2
#install.packages("invgamma")
library(invgamma)
mu <- seq(1.7, 1.9, 0.0001)
sigma2 <- seq(1/170, 1/50, 0.001)

apost <- matrix(NA, nrow = length(mu), ncol = length(sigma2))
for (i in 1:length(mu)) {
  for (j in 1:length(sigma2)) {
    apost[i, j] = dnorm(mu[i], mean = mu.n, sd = sqrt(sigma2[j]/kappa.n)) *
      dinvgamma(sigma2[j], shape=nu.n/2, rate=nu.n*sigma2.n/2)
  }
}

contour(mu,sigma2,apost)
```




```
par(mfrow=c(1,1))
```

3.3.2 Aposteriorna porazdelitev pri neinformativni apriorni porazdelitvi

Dobimo:

$$\mu \mid \sigma^2, x \sim N(\bar{x}, \sigma^2/n),$$

Aposteriorno porazdelitev σ^2 lahko zapisemo na tri ekvivalentne nacine, eden izmed njih:

$$(n-1)s^2/\sigma^2 \mid x \sim \chi^2_{n-1}.$$

Opazimo, da v aposteriorni porazdelitvi μ in σ^2 nista vec neodvisna, ceprav sta bila v apriorni.

Neinformativno porazdelitev si lahko predstavljamo nekako kot, da v konjugirani nastavimo κ_0 in ν_0 na 0.

Zgornji formuli sta dobro znani iz frekventisticne statistike.

3.4 Ali smo ocenili, kar nas je zanimalo?

Dejansko nas zanima robna (marginalna) aposteriorna porazdelitev μ.

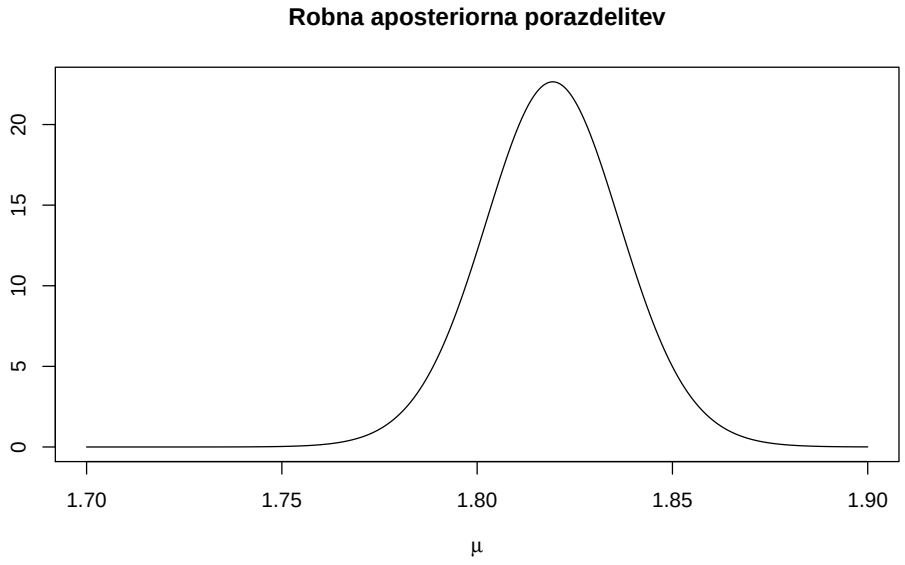
3.4.1 Robna aposterirona porazdelitev pri konjugirani apriorni porazdelitvi

V primeru konjugirane apriorne porazdelitve dobimo posploseno/nestandardizirano Studentovo porazdelitev:

$$\mu \mid x \sim t_{\nu_n}(\mu_n, \sigma_n^2/\kappa_n).$$

Narisemo v R-u:

```
mu <- seq(1.7, 1.9, 0.0001)
aposteriorna <- dt((mu-mu.n)/sqrt(sigma2.n/kappa.n), df = nu.n)/sqrt(sigma2.n/kappa.n)
plot(mu, aposteriorna, type="l", main="Robna aposteriorna porazdelitev",
     xlab = expression(mu), ylab = "")
```



3.4.2 Robna aposterirona porazdelitev pri neinformativni apriorni porazdelitvi

V primeru neinformativne apriorne porazdelitve dobimo posploseno/nestandardizirano Studentovo porazdelitev:

$$\mu \mid x \sim t_{n-1}(\bar{x}, s^2/n).$$

Oglejmo si, kaj velja za standardizirano odstopanje vzorcnega povprecja od populacijskega povprecja v naslednjih dveh pristopih:

- Bayesova statistika pri neinformativni apriorni porazdelitvi: $(\mu - \bar{x})/(s^2/n) \mid x \sim t_{n-1}$
- Frekventisticna statistika: $(\bar{x} - \mu)/(s^2/n) \mid \mu \sim t_{n-1}$

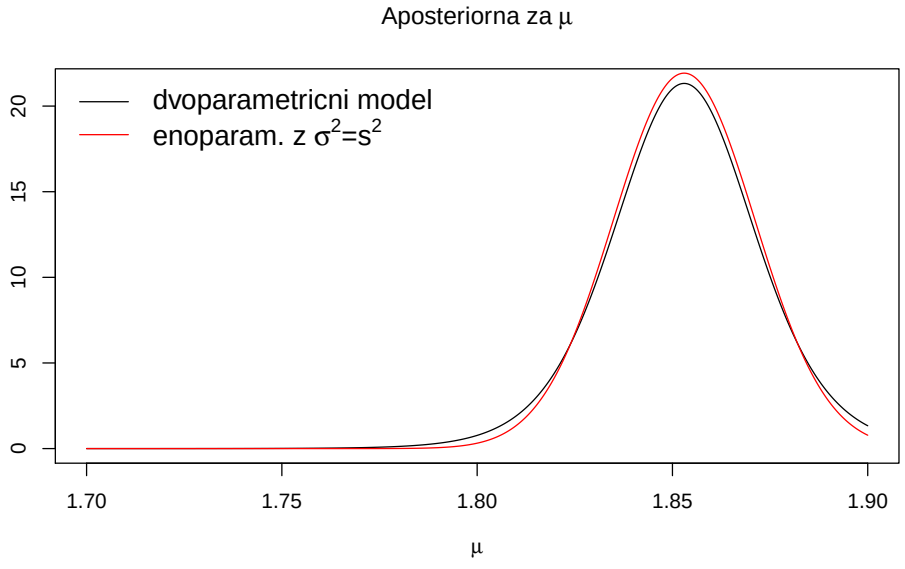
Negotovost o standardiziranemu odstopanju vzorcnega povprecja od populacijskega povprecja torej predstavlja enaka porazdelitev (tj. t_{n-1}) preden poznamo podatke (frekventisticna statistika) in po znanih podatkih (Bayesova statistika). Razlika pri tem je, da preden poznamo podatke dejansko ne poznamo niti $\bar{x}$ niti μ, po znanih podatkih pa poznamo $\bar{x}$, ki nam informira vedenje o μ.

Ce imamo v modelu $N(\mu, \sigma^2 = 0.1^2)$ neinformativno aposteriorno porazdelitev, potem je

$$\mu \mid x \sim N(\bar{x}, \sigma^2/n) = N(\bar{x}, 0.1^2/n).$$

Primerjamo rezultata iz obeh modelov (neinformativna apriorna v eno- ali dvoparametricnem modelu), kjer se zozimo na podvzorec prvih 10 visin (pri vecjem n razlika ne bi bila tako ocitna) in v modelu z znano varianco vzamemo kar $\sigma^2 = s^2$ (s cimer goljufamo, saj iz vzorca ocenimo varianco, ki naj bi bila znana vnaprej):

```
x.podvzorec <- x[1:10]

mu <- seq(1.7, 1.9, 0.0001)
aposteriorna.neinf <- dt((mu-mean(x.podvzorec))/sqrt(var(x.podvzorec)/n),
                         df = length(x.podvzorec)-1)/sqrt(var(x.podvzorec)/n)
aposteriorna.neinf.sigmaZnan <- dnorm(mu, mean = mean(x.podvzorec),
                                      sd = sqrt(var(x.podvzorec)/n))
plot(mu, aposteriorna.neinf, type="l",
     main=expression(paste("Aposteriorna za ",mu,sep="")),
     xlab = expression(mu), ylab = "")
lines(mu, aposteriorna.neinf.sigmaZnan, col="red")
legend("topleft", legend = c("dvoparametricni model",
       expression(paste("enoparam. z ",sigma^2,"=",s^2,sep=""))),
       col = c("black","red"), lty = 1, bty = "n", cex = 1.3)
```



Poanta: Ce goljufamo, potem je nasa ocena za μ manj variabilna – premalo variabilna!

3.5 Napovedovanje

Zanima nas, kaj lahko povemo o visini novega studenta ob upostevanju podatkov 30 studentov, tj. zanima nas **aposteriorna napovedna porazdelitev**.

V primeru konjugirane apriorne porazdelitve dobimo posploseno/nestandardizirano Studentovo porazdelitev:

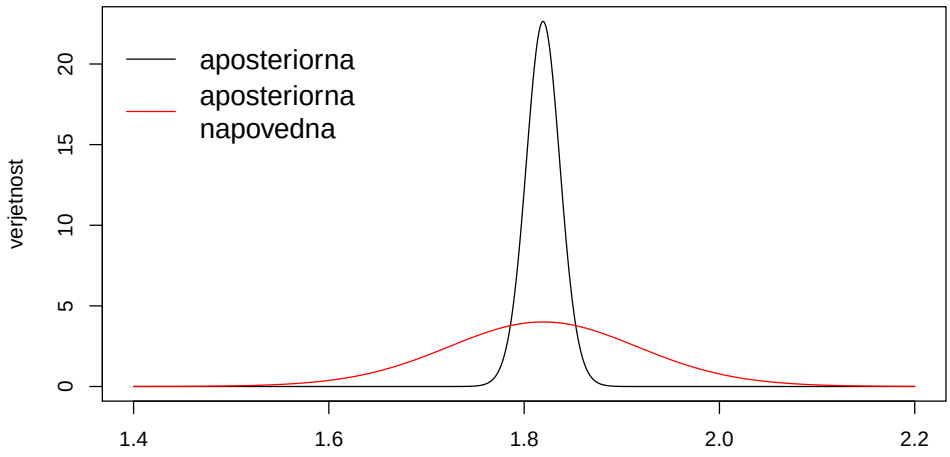
$$x_{nov} \mid x \sim t_{\nu_n}(\mu_n, \sigma_n^2/\kappa_n + \sigma_n^2).$$

V primeru neinformativne apriorne porazdelitve dobimo posploseno/nestandardizirano Studentovo porazdelitev:

$$x_{nov} \mid x \sim t_{n-1}(\bar{x}, s^2/n + s^2).$$

Poudarimo bistveno razliko med aposteriorno porazdelitvijo povprecne visine in aposteriorno napovedno porazdelitvijo za visino novega studenta (vzamemo primer s konjugirano apriorno porazdelitvijo):

```
mu <- seq(1.4, 2.2, 0.001)
aposteriorna <- dt((mu-mu.n)/sqrt(sigma2.n/kappa.n), df = nu.n)/sqrt(sigma2.n/kappa.n)
napovedna <- dt((mu-mu.n)/sqrt(sigma2.n/kappa.n+sigma2.n),
                df = nu.n)/sqrt(sigma2.n/kappa.n+sigma2.n)
plot(mu, aposteriorna, type="l",
     xlab = "", ylab = "verjetnost")
lines(mu, napovedna, col="red")
legend("topleft",  lty = 1,
       c("aposteriorna", "aposteriorna\nnapovedna"), col = c("black","red"),
       bty = "n", cex = 1.3)
```

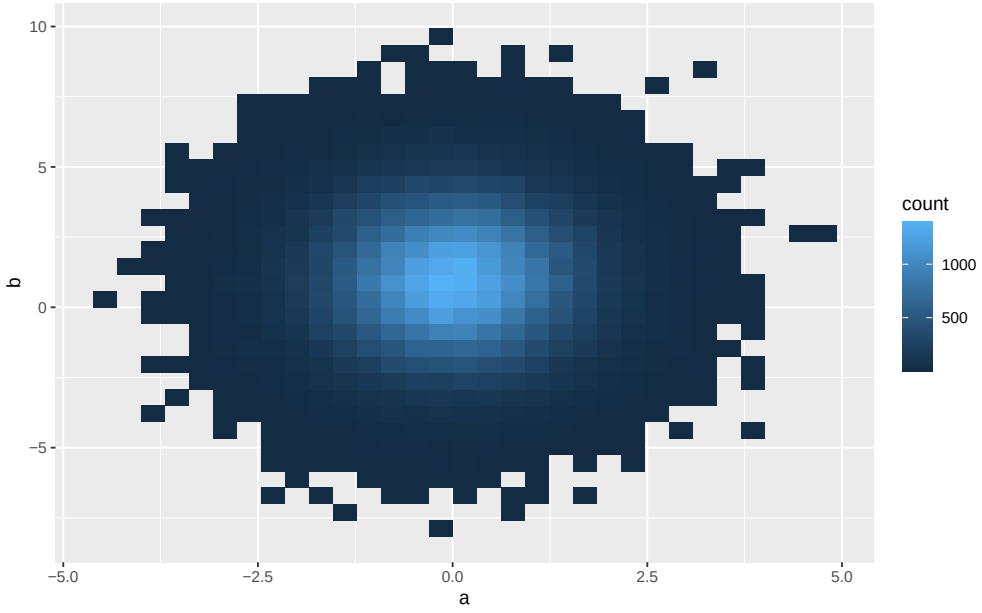


4 Naloga

Simulirajte iz aposteriorne porazdelitve, ki jo dobimo za nas primer pri zgoraj izbrani konjugirani apriorni porazdelitvi.

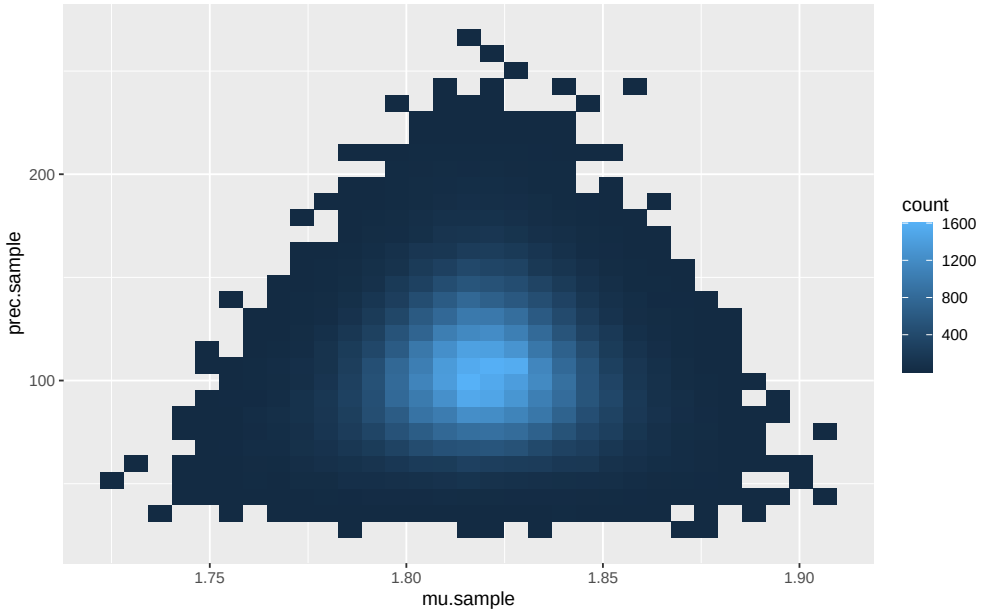
Za dobljeni vzorec narisite histrogram s knjiznico ggplot in ukazom stat_bin2d. Spodaj je primer za simulacijo iz bivariatne normalne porazdelitve, katere robni porazdelitvi sta neodvisni.

```
#install.packages("ggplot2")
library(ggplot2)
a <- rnorm(100000, 0, 1)
b <- rnorm(100000, 1, 2)
simulacija <- data.frame(a, b)
ggplot(simulacija, aes(a, b)) + stat_bin2d()
```

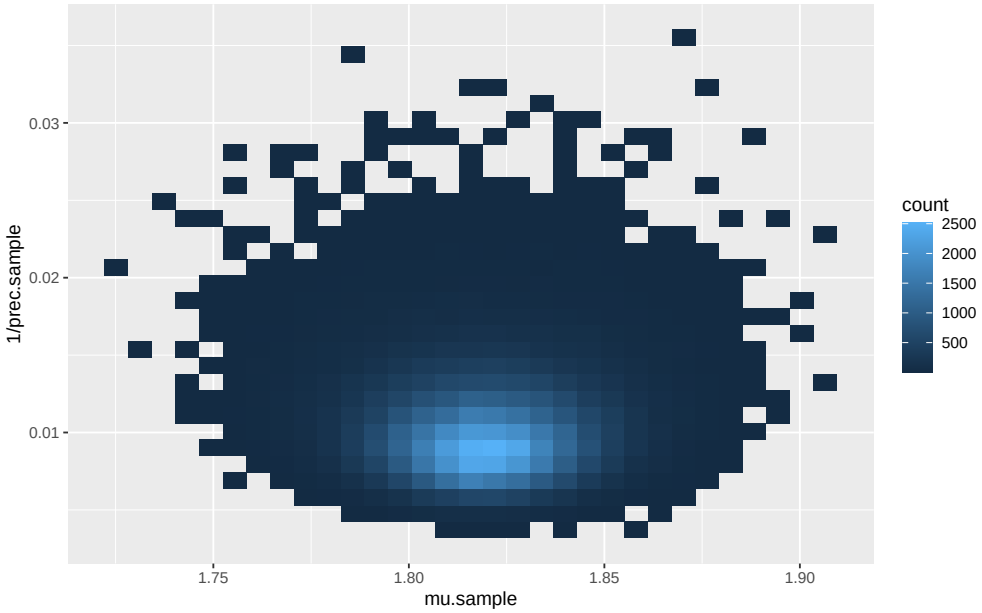


Resitev:

```
prec.sample <- rgamma(100000, nu.n/2, nu.n*sigma2.n/2)
mu.sample <- rnorm(100000, mu.n, sd = sqrt(1/(kappa.n*prec.sample)))

pod <- data.frame(mu.sample, prec.sample)
ggplot(pod, aes(mu.sample, prec.sample)) + stat_bin2d()
```




```
ggplot(pod, aes(mu.sample, 1/prec.sample)) + stat_bin2d()
```



6. sklop: Primer hierarhicnega modela z Gibbsovim vzorcevalnikom

Gradivo tega sklopa temelji na knjigi P. D. Hoff, *A First Course in Bayesian Statistical Methods*, 2009, in sicer na 8. poglavju *Group comparisons and hierarchical modeling*, str. 125.

1 Podatki

V raziskavi *Educational Longitudinal Study (ELS)* iz leta 2002 so preucevali rezultate testov ucencev ameriskih srednjih sol. Podane imamo rezultate matematicnih testov ucencev 10. razreda iz 100 javnih srednjih sol (velike sole z vsaj 400 ucenci 10. razreda, urbano okolje).

Za vsakega ucenca imamo podano solo in rezultat matematicnega testa – nasi podatki so torej vecnivojski/hierarhicni.

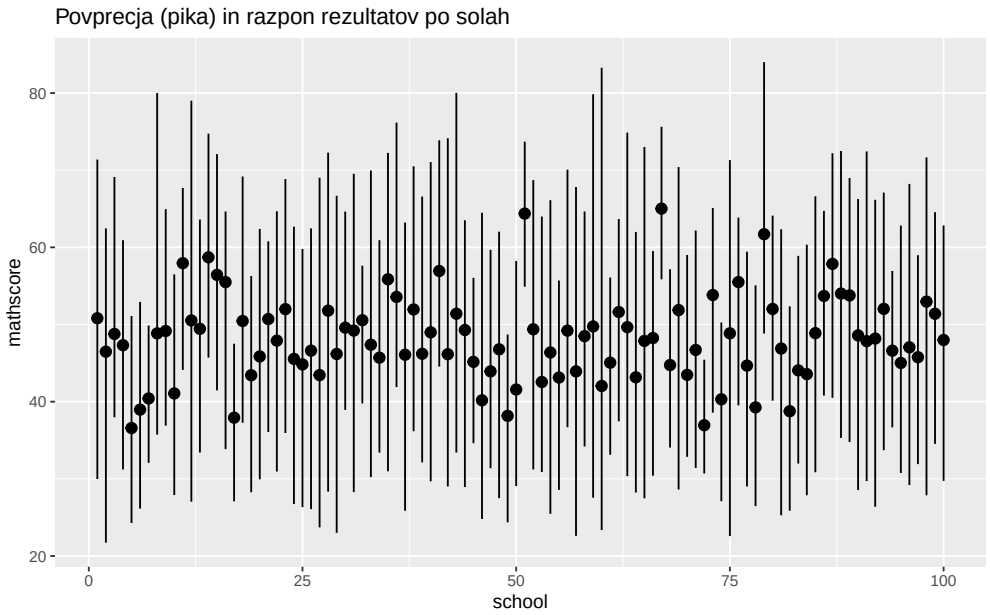
Rezultati matematicnega testa so del nacionalnega preverjanja znanja, ki naj bi bil konstruiran tako, da je pricakovana vrednost enaka 50 in standardni odklon 10.

Oglejmo si podatke.

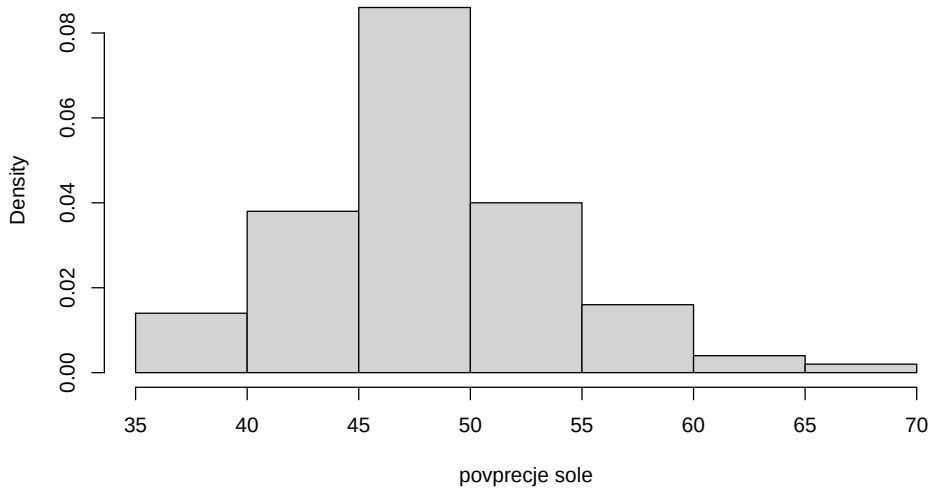
```
source("podatki_sole.R")
str(pod)
```

```
## 'data.frame':    1993 obs. of  2 variables:
##  $ school   : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ mathscore: num  52.1 57.6 66.4 44.7 40.6 ...
```

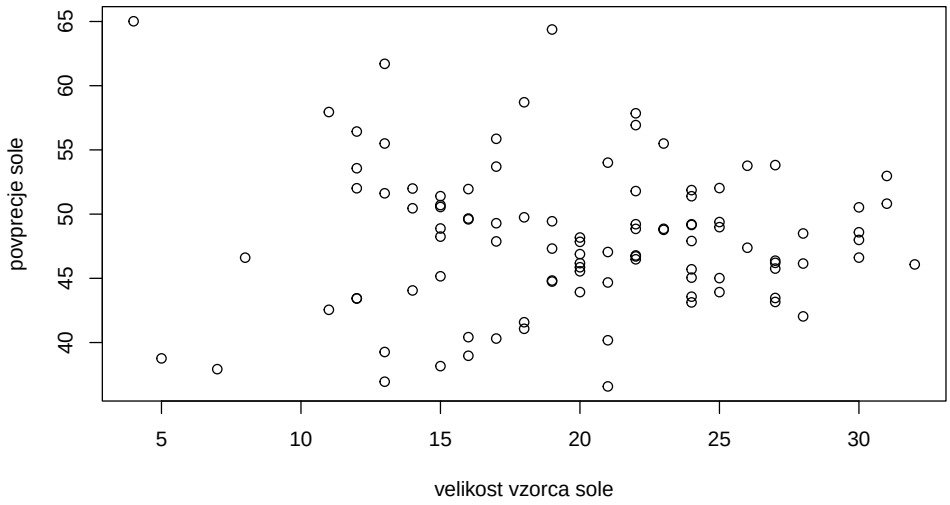
```
library(ggplot2)
ggplot(pod, aes(x = school, y = mathscore, group = school)) +
  stat_summary(fun.ymin = min, fun.ymax = max, fun.y = mean) +
  labs(title = "Povprecja (pika) in razpon rezultatov po solah")
```




```
library(dplyr)
pod.sole = pod %>%
  group_by(school) %>%
  summarise(povprecje = mean(mathscore), n=length(mathscore), varianca = var(mathscore))

hist(pod.sole$povprecje, prob=T,
     xlab = "povprecje sole", main = "")
```



```r
plot(pod.sole$n, pod.sole$povprecje,
     xlab = "velikost vzorca sole", ylab = "povprecje sole")
```



2 Model

2.1 Splosna ideja hierarhicnega modela

Podatki X_{ij}, $i \in \{1, 2, \ldots, n_j\}$, $j \in \{1, 2, \ldots, m\}$ (m skupin, velikost posamezne skupine n_j).

Variabilnost znotraj skupine (*within-group sampling variability*) – model:

$$(X_{1,j}, \ldots, X_{n_j,j}) \mid \theta_j \sim \text{n.e.p. } f(x \mid \theta_j)$$

Variabilnost med skupinami (*between-group sampling variability*) – model in apriorne porazdelitve θ_j:

$$(\theta_1, \ldots, \theta_m) \mid \phi \sim \text{n.e.p. } f(\theta \mid \phi)$$

Bistvo hierarhicnega modela: podatki so odvisni od parametrov le preko θ_j in ne tudi ϕ (tj. $f(x \mid \theta_j, \phi) = f(x \mid \theta_j)$), θ_j pa so realizacija vzorca (tj. n.e.p.) iz neke skupne porazdelitve s hiperparametrom. Okrajsava: n.e.p. = neodvisne enako porazdeljene (iid).

Apriorna, oziroma natancneje *hiperapriorna* porazdelitev za ϕ:

$$\phi \sim \pi(\phi)$$

V modelu imamo *parametre* $\theta_1, \ldots, \theta_m$ (vsak je lahko vektor) in *hiperparametre* ϕ (lahko vektor).

2.2 Hierarhicni normalni model

Variabilnost znotraj skupine (*within-group sampling variability*) – model:

$$(X_{1,j}, \ldots, X_{n_j,j}) \mid \mu_j, \sigma^2 \sim \text{n.e.p. } N(\mu_j, \sigma^2)$$

Variabilnost med skupinami (*between-group sampling variability*) – model in apriorne porazdelitve μ_j:

$$(\mu_1, \ldots, \mu_m) \mid \mu, \eta^2 \sim \text{n.e.p. } N(\mu, \eta^2)$$

Privzeli smo, da so vse variance znotraj skupin enake (σ^2).

Potrebujemo torej se (hiper)apriorno porazdelitev za (μ, η^2) in apriorno proazdelitev σ^2.

Izberemo si:

$$\sigma^2 \sim \text{Inv-Gama}(\nu_0/2, \sigma_0^2\nu_0/2).$$

Za hiperapriorno porazdelitev vzamemo "polkonjugirano" (te nismo obravnavali v 5. sklopu, glejte 6. poglavje v knjigi Hoff):

$$\begin{aligned} \pi(\mu, \eta^2) &= \pi(\mu)\,\pi(\eta^2), \\ \mu &\sim \mathcal{N}(\mu_0, \tau_0^2), \\ \eta^2 &\sim \text{Inv-Gama}(\kappa_0/2, \eta_0^2\kappa_0/2). \end{aligned}$$

Zapisali smo matematicni model.

Za boljso predstavo si narisemo graficni model (vaje).

2.3 Apriorna porazdelitev – natancneje

Imamo torej naslednjo vecrazsezno apriorno porazdelitev (upostevamo zgornje izbore gostot in kateri parametri so neodvisni):

$$\begin{aligned} \pi(\mu_1, \ldots, \mu_m, \mu, \eta^2, \sigma^2) &= \pi(\mu_1, \ldots, \mu_m \mid \mu, \eta^2) \cdot \pi(\sigma^2) \cdot \pi(\mu, \eta^2) \\ &= \left(\prod_{j=1}^{m} \pi(\mu_j \mid \mu, \eta^2) \right) \cdot \pi(\sigma^2) \cdot \pi(\mu) \cdot \pi(\eta^2) \\ &= \left(\prod_{j=1}^{m} f_{\text{Normal}}(\mu_j \mid \mu, \eta^2) \right) \cdot f_{\text{Inv-Gama}}(\sigma^2 \mid \sigma_0^2, \nu_0) \cdot f_{\text{Normal}}(\mu \mid \mu_0, \tau_0^2) \\ &\quad \cdot f_{\text{Inv-Gama}}(\eta^2 \mid \eta_0^2, \kappa_0), \end{aligned}$$

kjer so $\mu_0, \tau_0^2, \sigma_0^2, \nu_0, \eta_0^2, \kappa_0$ konstante.

2.4 Aposteriorna porazdelitev

Zelimo vzorciti iz aposteriorne porazdelitve

$$\pi(\mu_1, \ldots, \mu_m, \mu, \eta^2, \sigma^2 \mid [x_{ij}]_{ij}) = \pi(\mu_1, \ldots, \mu_m, \mu, \eta^2, \sigma^2 \mid \boldsymbol{x}_1, \ldots, \boldsymbol{x}_m).$$

Ne bomo izpeljali/podali cele vecrazsezne gostote (kot v sklopih 1-3 in 5), uporabili bomo Gibbsov vzorcevalnik.

2.5 Gibbsov vzorcevalnik – splosno

Vzorciti zelimo iz aposteriorne porazdelitve

$$\pi(\theta, \phi \mid x).$$

Za Gibbsov algoritem moramo poznati obe pogojni aposteriorni porazdelitvi:

$$\pi(\theta \mid \phi, x),$$

$$\pi(\phi \mid \theta, x).$$

Na koraku s Gibbsovega vzorcevalnika smo dobili $(\theta^{(s)}, \phi^{(s)})$. Naslednji korak:

1. Vzorcimo $\theta^{(s+1)} \sim \pi(\theta \mid \phi^{(s)}, x)$.
2. Vzorcimo $\phi^{(s+1)} \sim \pi(\phi \mid \theta^{(s+1)}, x)$.

2.6 Aposteriorna porazdelitev - nadaljevanje

Potrebujemo torej vse pogojne aposteriorne porazdelitve.

Najprej razpisimo aposteriorno porazdelitev, podobno kakor smo apriorno:

$$\begin{aligned}
&\pi(\mu_1, \ldots, \mu_m, \mu, \eta^2, \sigma^2 \mid \boldsymbol{x}_1, \ldots, \boldsymbol{x}_m) \\
&\propto \pi(\mu_1, \ldots, \mu_m, \mu, \eta^2, \sigma^2) \cdot f(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m \mid \mu_1, \ldots, \mu_m, \mu, \eta^2, \sigma^2) \\
&= \pi(\mu, \eta^2, \sigma^2) \cdot \pi(\mu_1, \ldots, \mu_m \mid \mu, \eta^2, \sigma^2) \cdot f(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m \mid \mu_1, \ldots, \mu_m, \sigma^2) \\
&= \pi(\mu, \eta^2) \cdot \pi(\sigma^2) \cdot \pi(\mu_1, \ldots, \mu_m \mid \mu, \eta^2) \cdot f(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m \mid \mu_1, \ldots, \mu_m, \sigma^2) \\
&= \pi(\mu) \cdot \pi(\eta^2) \cdot \pi(\sigma^2) \cdot \left[\prod_{j=1}^{m} \pi(\mu_j \mid \mu, \eta^2) \right] \cdot \left[\prod_{j=1}^{m} \prod_{i=1}^{n_j} f(x_{i,j} \mid \mu_j, \sigma^2) \right) \Bigg].
\end{aligned}$$

Velja torej:

$$\begin{aligned}
\pi(\mu \mid \mu_1, \ldots, \mu_m, \eta^2, \sigma^2, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_m) &\propto \pi(\mu) \prod_{j=1}^{m} \pi(\mu_j \mid \mu, \eta^2) \\
&= f_{\text{Normal}}(\mu \mid \mu_0, \tau_0^2) \cdot \left(\prod_{j=1}^{m} f_{\text{Normal}}(\mu_j \mid \mu, \eta^2) \right).
\end{aligned}$$

Iz tega se ne znamo vzorciti, izracunati moramo gostoto. Opazimo lahko, da je to aposteriorna porazdelitev za parameter μ iz normalnega modela z znano varianco η^2, kjer ima $(\mu_1, ..., \mu_m)$ vlogo vzorca velikosti m, in ima μ konjugirano apriorno porazdelitev $N(\mu_0, \tau_0^2)$. Po formulah iz 3. sklopa (razdelek 4.3, str. 5) je torej

$$\mu \mid \text{vse ostalo} \sim N\left(\frac{\bar{\mu} m/\eta^2 + \mu_0/\tau_0^2}{m/\eta^2 + 1/\tau_0^2}, \left[m/\eta^2 + 1/\tau_0^2 \right]^{-1} \right).$$

Podobno lahko izpeljemo se preostale pogojne aposteriorne porazdelitve, kjer uporabimo ze znano iz (1) normalnega modela z znano varianco in konjugirano apriorno porazdelitvijo za povprecje; in (2) dvoparametricnega normalnega modela s polkonjugirano apriorno porazdelitvijo (v 5. sklopu smo imeli konjugirano, vec o polkonjugirani pa v 6. poglavju knjige Hoff). Dobimo:

$$\eta^2 \mid \text{vse ostalo} \sim \text{Inv-Gama} \quad \frac{\kappa_0 + m}{2}, \frac{\kappa_0\eta_0^2 + \sum_{j=1}^{m}(\mu_j - \mu)^2}{2}\Bigg)$$
$$\mu_j \mid \text{vse ostalo} \sim N\left(\frac{\bar{x}_{.j} n_j/\sigma^2 + \mu/\eta^2}{n_j/\sigma^2 + 1/\eta^2}, \left[n_j/\sigma^2 + 1/\eta^2\right]^{-1}\right)$$
$$\sigma^2 \mid \text{vse ostalo} \sim \text{Inv-Gama} \quad \frac{\nu_0 + \sum_{j=1}^{m} n_j}{2}, \frac{\nu_0\sigma_0^2 + \sum_{j=1}^{m}\sum_{i=1}^{n_j}(x_{i,j} - \mu_j)^2}{2}\Bigg)$$

3 Uporaba modela na podatkih

Dolociti moramo se parametre (hiper)apriornih porazdelitev $\mu_0, \tau_0^2, \sigma_0^2, \nu_0, \eta_0^2, \kappa_0$:

$$\begin{aligned}\sigma^2 &\sim \text{Inv-Gama}(\nu_0/2, \sigma_0^2\nu_0/2),\\ \mu &\sim \mathcal{N}(\mu_0, \tau_0^2),\\ \eta^2 &\sim \text{Inv-Gama}(\kappa_0/2, \eta_0^2\kappa_0/2).\end{aligned}$$

Spomnimo se, da je matematicni test konstruiran tako, da je pricakovana vrednost enaka 50 in standardni odklon 10.

Izberemo si:
$\sigma_0^2 = 100$ (ker naj bi bil standardni odklon 10), $\nu_0 = 1$ (sibko informativna),
$\mu_0 = 50$ (ker naj bi bil standardni odklon 10),
$\tau_0^2 = 25$ (ker zelimo sibko informativno in s tem dopuscamo s 95% μ med 40 in 60),
$\eta_0^2 = 100$, $\kappa_0 = 1$.

3.1 Gibbsov vzorcevalnik za nas primer

```
### Parametri (hiper)apriornih porazdelitev
sigma20 = 100
nu0 = 1
eta20 = 100
kappa0 = 1
mu0 = 50
tau20 = 25

### Pripravimo si kolicine, ki jih bomo potrebovali iz podatkov
x = pod
m = length(pod.sole$school)
n = pod.sole$n
x.povpr = pod.sole$povprecje

### Dolocimo si zacetne vrednosti
muGroups = x.povpr
sigma2 = mean(pod.sole$varianca)
mu = mean(muGroups)
eta2 = var(muGroups)

### Pripravimo si prostor za shranjevanje
n.iter = 5000

muGroups.all = matrix(nrow = n.iter, ncol = m)
sigma2.all = rep(NA, n.iter)
mu.all = rep(NA, n.iter)
eta2.all = rep(NA, n.iter)
```

```
### Na prvo mesto si shranimo zacetne vrednosti (nepotrebno)
muGroups.all[1, ] = muGroups
sigma2.all[1] = sigma2
mu.all[1] = mu
eta2.all[1] = eta2

### Pozenemo Gibbsov vzorcevalnik
set.seed(1)
for (s in 1:n.iter) {
  ### Vzorcimo muGroups
  for (j in 1:m) {
    muGroups[j] = rnorm(1,
                        mean = (x.povpr[j] * n[j] / sigma2 + mu / eta2) /
                          (n[j] / sigma2 + 1 / eta2),
                        sd = sqrt(1 / (n[j] / sigma2 + 1 / eta2)))
  }

  ### Vzorcimo sigma2
  ss = nu0 * sigma20
  for (j in 1:m) {
    ss = ss + sum((x[x[, 1] == j, 2] - muGroups[j])^2)
  }
  sigma2 = 1 / rgamma(1, (nu0 + sum(n)) / 2, ss / 2)

  ### Vzorcimo mu
  mu = rnorm(1,
             mean = (mean(muGroups) * m / eta2 + mu0 / tau20) /
               (m / eta2 + 1 /tau20),
             sd = sqrt(1 / (m / eta2 + 1 /tau20)))

  ### Vzorcimo eta2
  ss = kappa0 * eta20 + sum((muGroups - mu)^2)
  eta2 = 1 / rgamma(1, (kappa0 + m) / 2, ss / 2)

  ### Shranimo nove parametre
  muGroups.all[s, ] = muGroups
  sigma2.all[s] = sigma2
  mu.all[s] = mu
  eta2.all[s] = eta2
}
```

3.2 Preucevanje konvergence

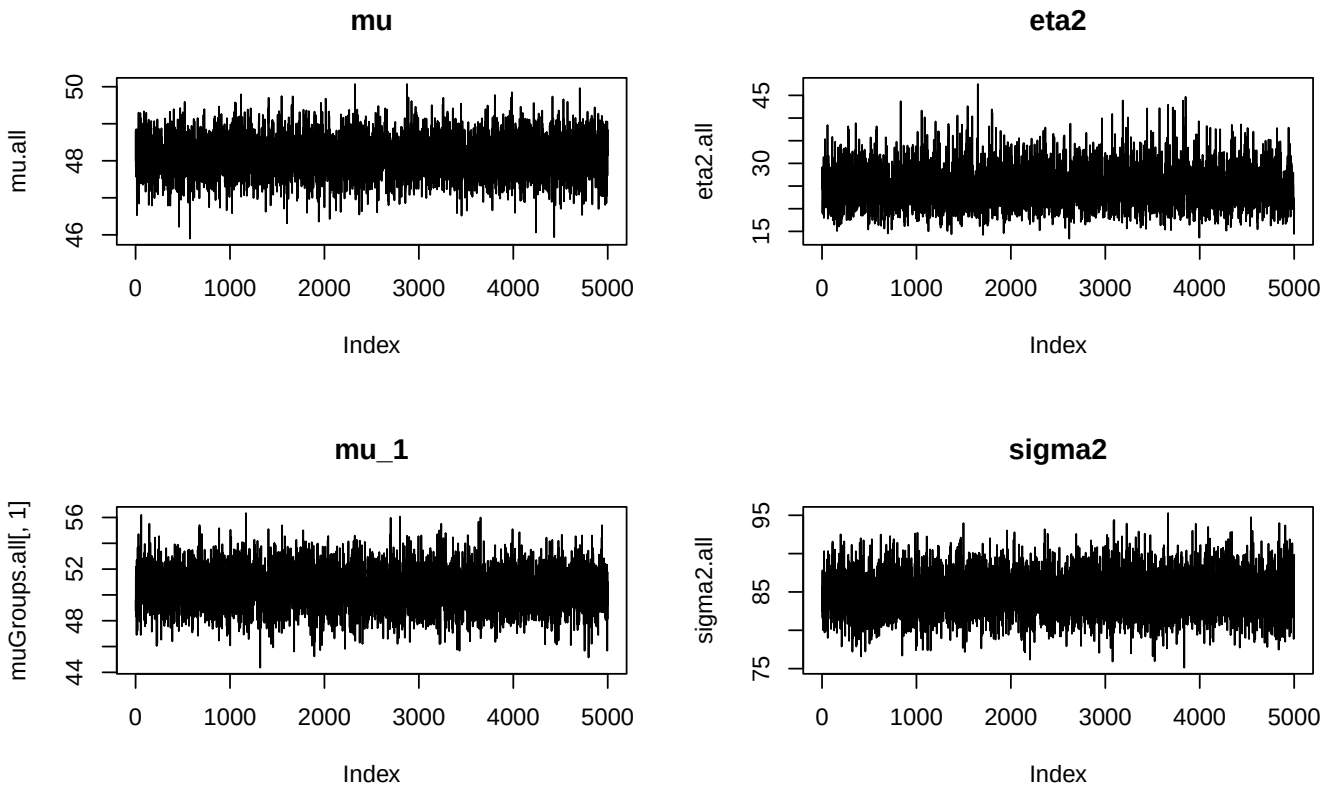
V podrazdelkih so navedeni nacini za preucevanje konvergence.

3.2.1 *Trace plots*

Na podlagi spodnjih slik verig dolocimo potreben *burn-in* in graficno presodimo, ali je konvergenca dosezena (tako kot v 4. sklopu).

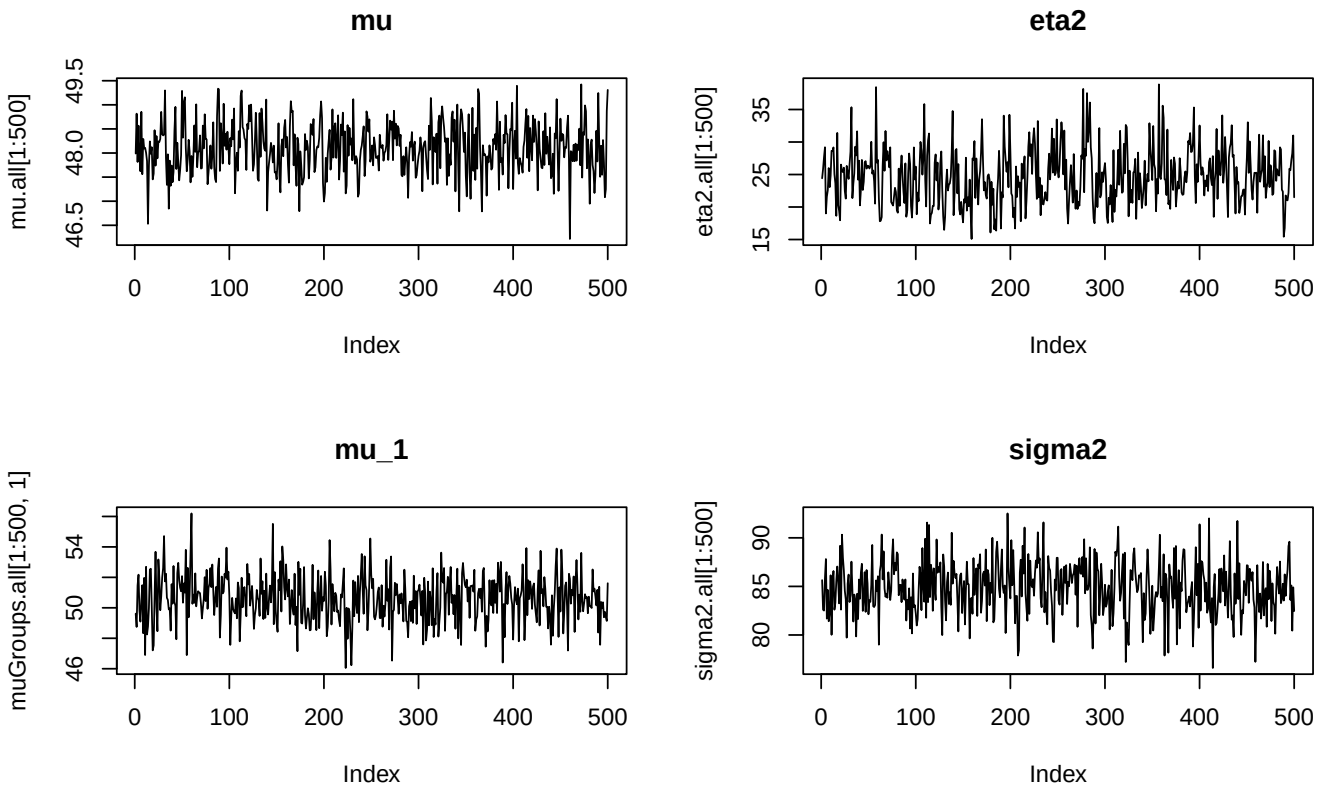
Vse iteracije (izgleda v redu):

```
par(mfrow=c(2,2))
plot(mu.all, type="l", main="mu")
plot(eta2.all, type="l", main="eta2")
plot(muGroups.all[,1], type="l", main="mu_1")
plot(sigma2.all, type="l", main="sigma2")
```



Prvih 500 iteracij (izgleda v redu):

```
par(mfrow=c(2,2))
plot(mu.all[1:500], type="l", main="mu")
plot(eta2.all[1:500], type="l", main="eta2")
plot(muGroups.all[1:500,1], type="l", main="mu_1")
plot(sigma2.all[1:500], type="l", main="sigma2")
```



3.2.2 Vec verig in *mixing*

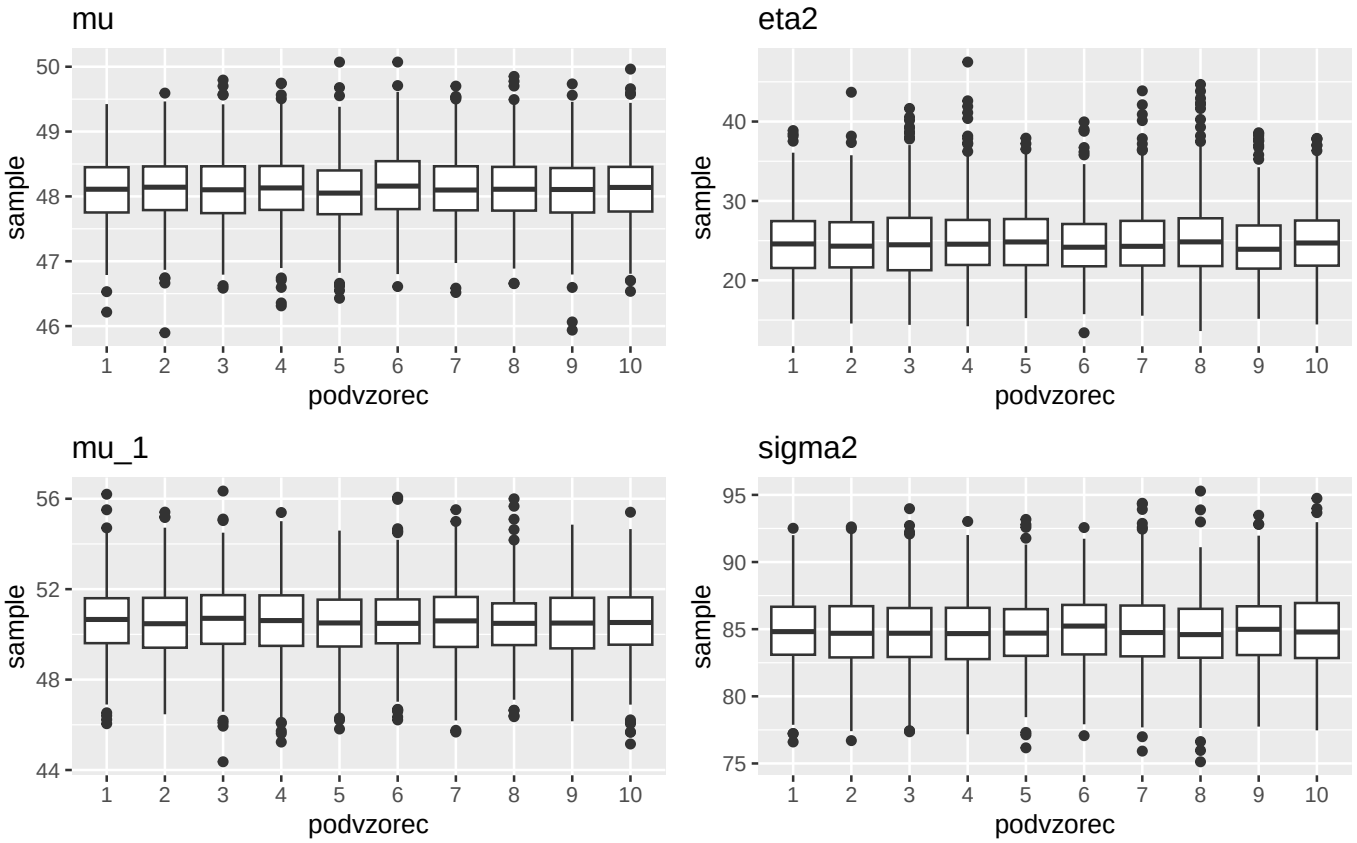
Preizkusimo razlicne zacetne vrednosti (kljucno je, da vkljucimo tudi manj smiselne) in pogledamo, ali dobimo po nekem zacetnem *burn-in* podobno verigo. Verigi lahko narisemo na eno sliko in pogledamo, ali je dober *mixing*. V koncni vzorec potem zdruzimo vse verige.

Tukaj analizo vecih verig izpustimo (to smo podrobneje preucevali ze pri algoritmu Metropolis-Hastings).

3.2.3 Porazdelitve podvzorcev

Pogledamo, ali se približno ujemajo porazdelitve po zaporednih odsekih, tj. ali je veriga "stabilna" (izgleda v redu).

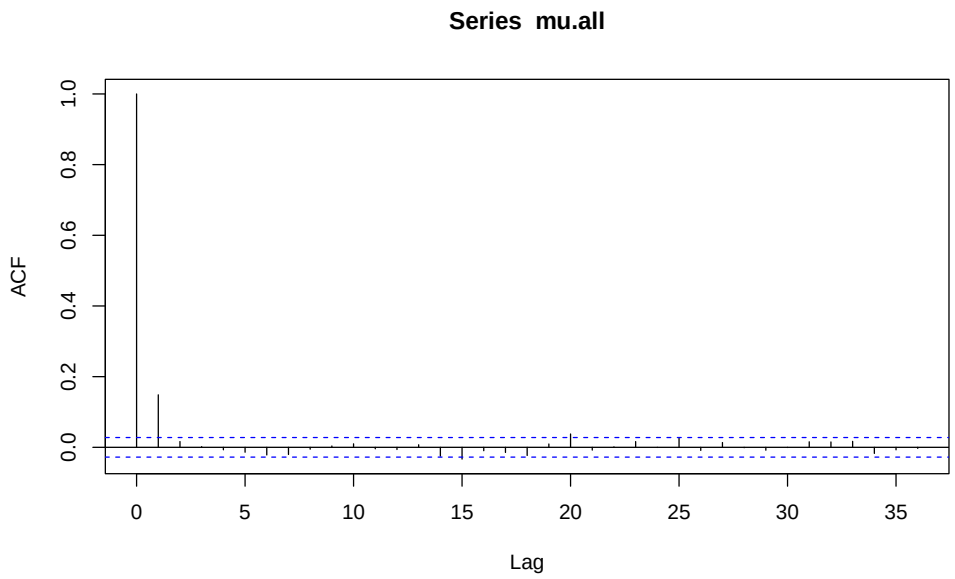
```
library(gridExtra)
mu.all2 = data.frame(sample = mu.all, podvzorec = factor(sort(rep(1:10,500))))
p1 = ggplot(mu.all2, aes(x = podvzorec, y = sample)) +
  geom_boxplot() + labs(title = "mu")

eta2.all2 = data.frame(sample = eta2.all, podvzorec = factor(sort(rep(1:10,500))))
p2 = ggplot(eta2.all2, aes(x = podvzorec, y = sample)) +
  geom_boxplot() + labs(title = "eta2")

mu1 = data.frame(sample = muGroups.all[,1], podvzorec = factor(sort(rep(1:10,500))))
p3 = ggplot(mu1, aes(x = podvzorec, y = sample)) +
  geom_boxplot() + labs(title = "mu_1")

sigma2.all2 = data.frame(sample = sigma2.all, podvzorec = factor(sort(rep(1:10,500))))
p4 = ggplot(sigma2.all2, aes(x = podvzorec, y = sample)) +
  geom_boxplot() + labs(title = "sigma2")
grid.arrange(p1, p2, p3, p4, ncol = 2)
```



3.2.4 Avtokorelacije in *effective sample size* VS *thinning*

Pri zamiku (*lag*) 1 dobimo po definiciji avtokorelacijo 1, za kasnejse pa si zelimo, da so cim blizje 0 (kar pricakujemo pri n.e.p. vzorcu). Pri vzorcu dobljenim z MCMC metodami bo prisotna avtokorelacija, ki pa se z zamikom (*lag*) zmanjsuje (odvisnost neke vrednosti od vrednosti iz enega koraka nazaj je najvecja, iz dveh korakov nazaj malo manjsa, itd.).

```
acf(mu.all) #dobimo graf, cel vzorec
```




```
ac.mu = acf(mu.all, plot = FALSE)
ac.mu #dobimo izpis
```

```
##
## Autocorrelations of series 'mu.all', by lag
##
##      0      1      2      3      4      5      6      7      8      9     10
##  1.000  0.149  0.016  0.002 -0.007 -0.014 -0.022 -0.020 -0.005  0.004  0.010
##     11     12     13     14     15     16     17     18     19     20     21
## -0.004 -0.006  0.007 -0.025 -0.033 -0.010 -0.015 -0.023  0.009  0.038 -0.008
##     22     23     24     25     26     27     28     29     30     31     32
##  0.001  0.016  0.000  0.026 -0.009  0.014 -0.001 -0.008 -0.001  0.016  0.015
##     33     34     35     36
##  0.017 -0.017 -0.007 -0.003
```

```
head(ac.mu$acf) #dobimo natancnejsi izpis
```

```
## , , 1
##
##              [,1]
## [1,]  1.000000000
```

```
## [2,]  0.148547488
## [3,]  0.016374954
## [4,]  0.001949737
## [5,] -0.007071617
## [6,] -0.013949257
```

Kako lahko priblizno izracunamo avtokorelacijo z zamikom 1?

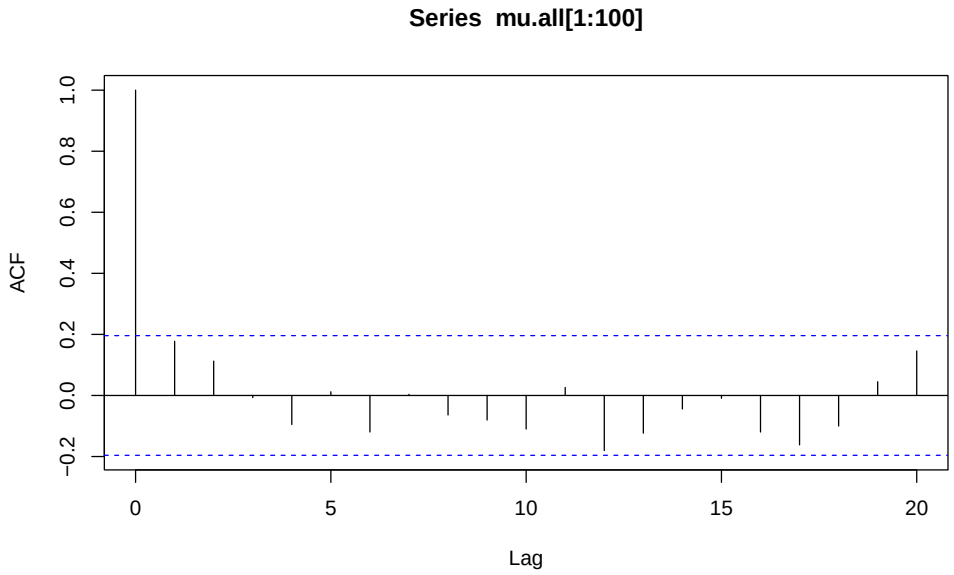
```
cor(mu.all[-length(mu.all)], mu.all[-1])
```

```
## [1] 0.1485619
```

```
ac.mu$acf[2]
```

```
## [1] 0.1485475
```

```
acf(mu.all[1:100]) #za prvih 100 iteracij
```

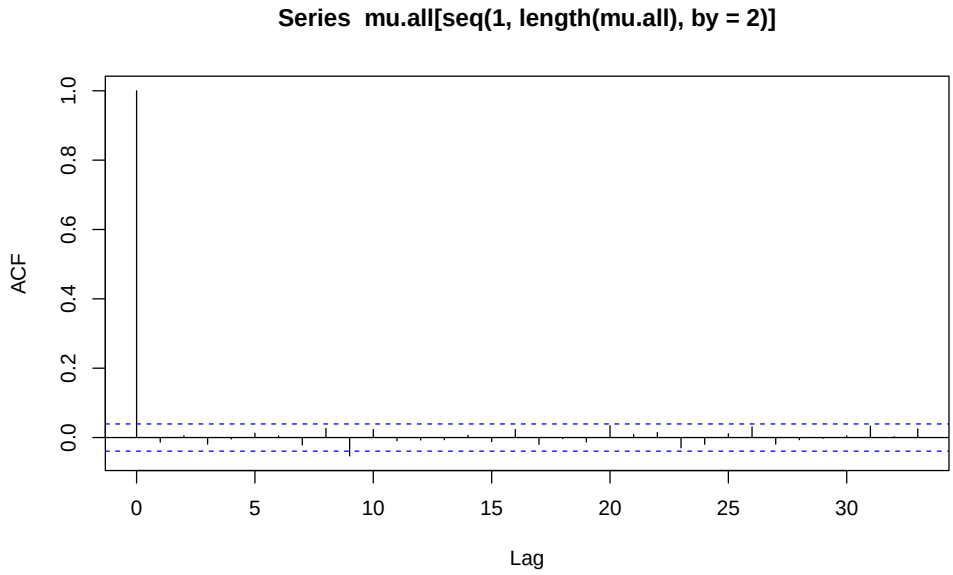


Kaj predstavljata crti?

$\pm 1.96/\sqrt{N}$, N stevilo iteracij – avtokorelacije izven obmocja so statisticno znacilno razlicne od 0 (oz. le 5% jih lahko pricakujemo *nekoliko* izven pri n.e.p. vzorcu).

Kaj se v nasem primeru zgodi z avtokorelacijami, ce v zaporedju izbrisemo vsakega drugega (uporabimo *thinning*, glejte 4. sklop)?

```
acf(mu.all[seq(1, length(mu.all), by=2)])
```



Pricakovano se zmanjsajo, vendar smo pri tem zmanjsali tudi velikost vzorca, za katerega smo ze porabili cas za izracun.

Effective sample size lahko interpretiramo kakor stevilo slucajnih vzorcev (n.e.p.), ki nam dajo enako natancno informacijo kakor nas dobljeni MCMC vzorec.

```
library(coda)
effectiveSize(mu.all)
```

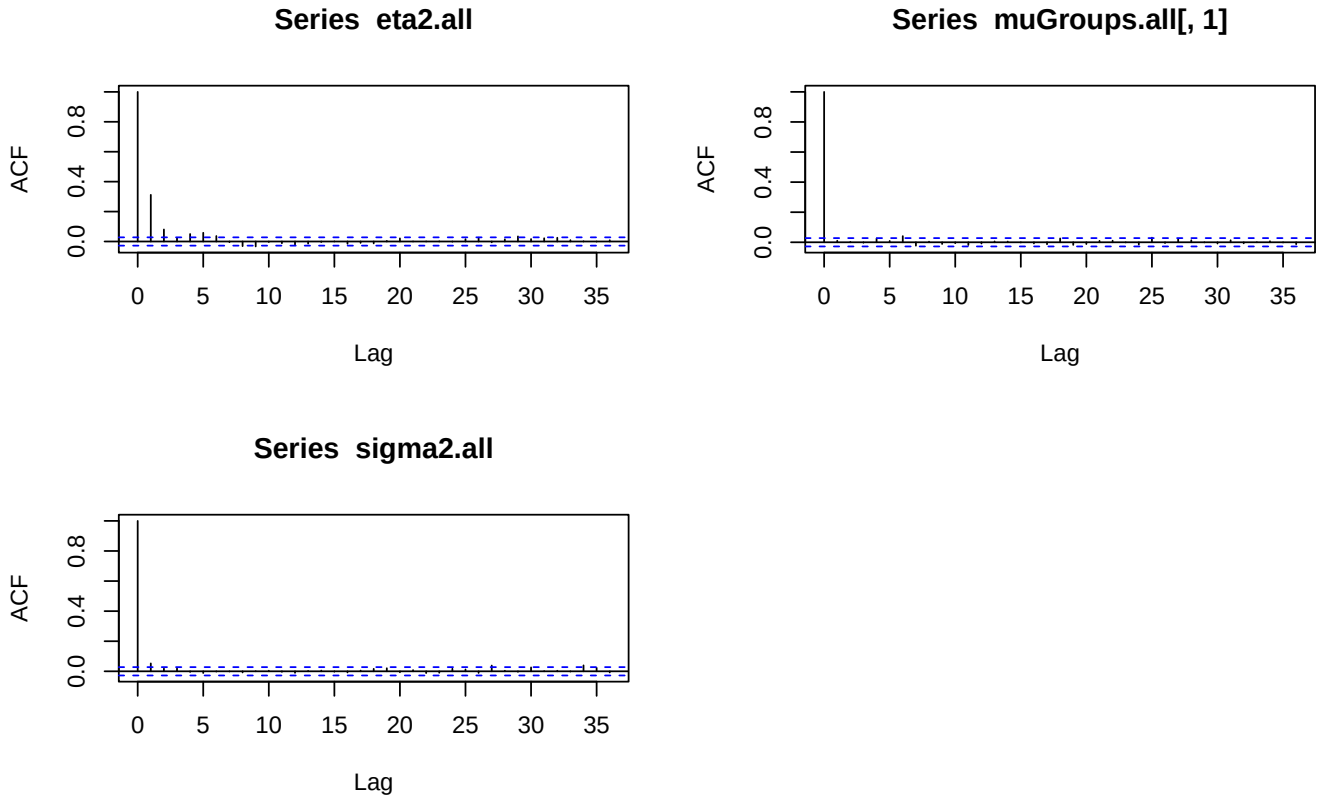
```
##     var1
## 3705.908
```

Torej:

- Celoten vzorec za μ dolzine 5000 je avtokoreliran – ne smemo ga imeti za slucajni vzorec (n.e.p.) iz aposteriorne porazdelitve.
- Ce v verigo vzamemo vsakega drugega (*thinning* $= 2$, kar je najmanjse mozno), potem vzorec ni avtokoreliran, njegova velikost pa je 2500.
- *Effective sample size*, ki pove za koliko n.e.p. realizacij je nas vzorec "vreden", je enak 3705, kar je mnogo vec od 2500. Uporabimo zato raje celoten vzorec in upostevamo *effective sample size* kot njegovo "pravo velikost".

Avtokorelacije preostalih (nekaj) parametrov in njihovi *effective sample size*:

```
par(mfrow=c(2,2))
acf(eta2.all)
acf(muGroups.all[,1])
acf(sigma2.all)
```




```
effectiveSize(eta2.all)
```

```
##     var1
## 2503.468
```

```
effectiveSize(muGroups.all[,1])
```

```
##     var1
## 4421.197
```

```
effectiveSize(sigma2.all)
```

```
##     var1
## 4498.545
```

Preko *effective sample size* izracunamo **"standardne napake" nasih ocen**, kjer je ocena povprecje marginalne aposteriorne porazdelitve:

```
sd(mu.all) / sqrt( effectiveSize(mu.all) )
```

```
##        var1
## 0.008795844
```

```
sd(eta2.all) / sqrt( effectiveSize(eta2.all) )
```

```
##       var1
## 0.08842911
```

```
sd(muGroups.all[,1]) / sqrt( effectiveSize(muGroups.all[,1]) )
```

```
##       var1
## 0.02359033
```

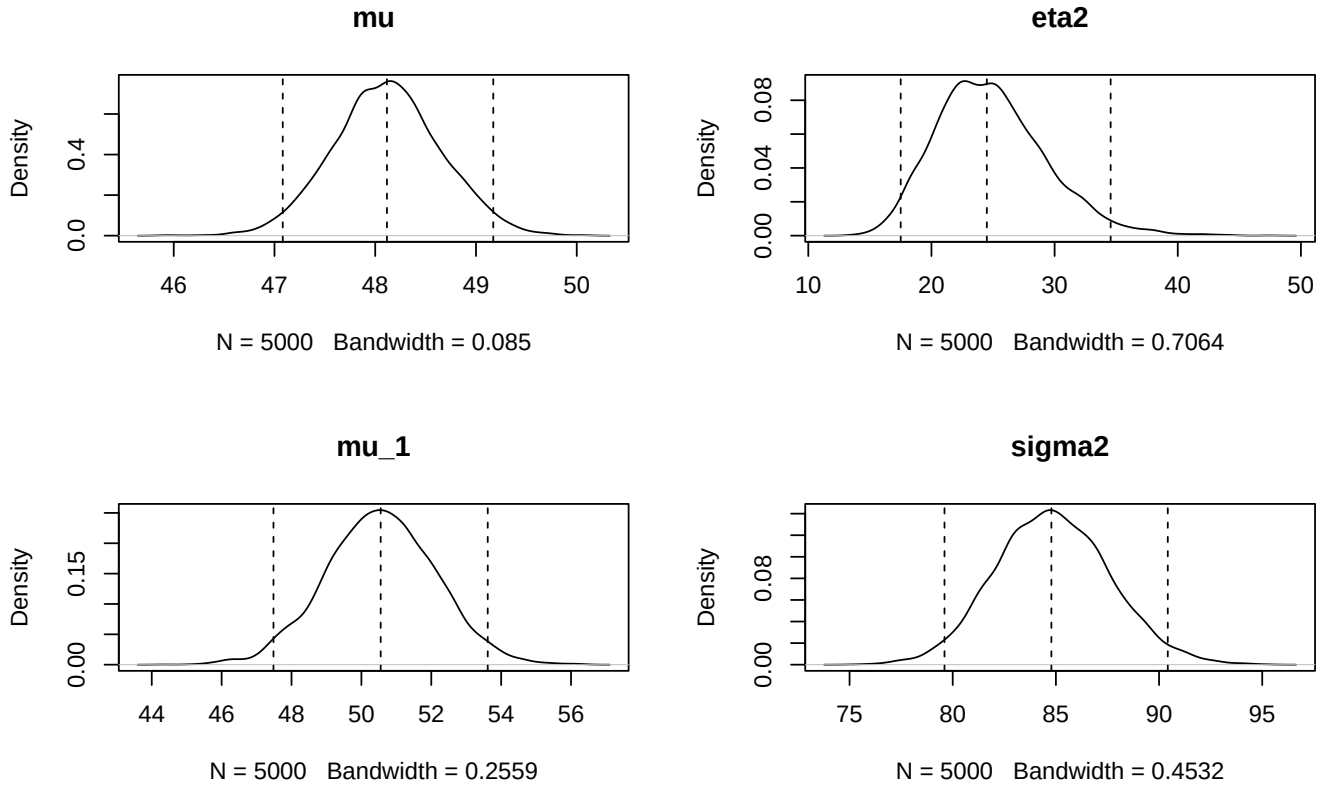
```
sd(sigma2.all) / sqrt( effectiveSize(sigma2.all) )
```

```
##       var1
## 0.04123463
```

Ali so velike ali majhne? Odvisno glede na skalo parametrov oz. enote spremenljivk.

3.3 Robne aposteriorne porazdelitve

```
par(mfrow=c(2,2))
plot(density(mu.all), type="l", main="mu")
abline(v = quantile(mu.all, prob=c(0.025, 0.5, 0.975)), lty = 2)
plot(density(eta2.all), type="l", main="eta2")
abline(v = quantile(eta2.all, prob=c(0.025, 0.5, 0.975)), lty = 2)
plot(density(muGroups.all[,1]), type="l", main="mu_1")
abline(v = quantile(muGroups.all[,1], prob=c(0.025, 0.5, 0.975)), lty = 2)
plot(density(sigma2.all), type="l", main="sigma2")
abline(v = quantile(sigma2.all, prob=c(0.025, 0.5, 0.975)), lty = 2)
```



Interpretirati rezultate (srednja vrednost in razprsenost porazdelitve sta pomembni) tako za hiperparametre, kakor tudi za posamezne sole (katera ima najboljse, katera najslabse rezultate, ali se parametri "statisticno znacilno" razlikujejo, ipd.).

Preveriti odnos med aposterornimi in apriornimi porazdelitvami (hiper)parametrov

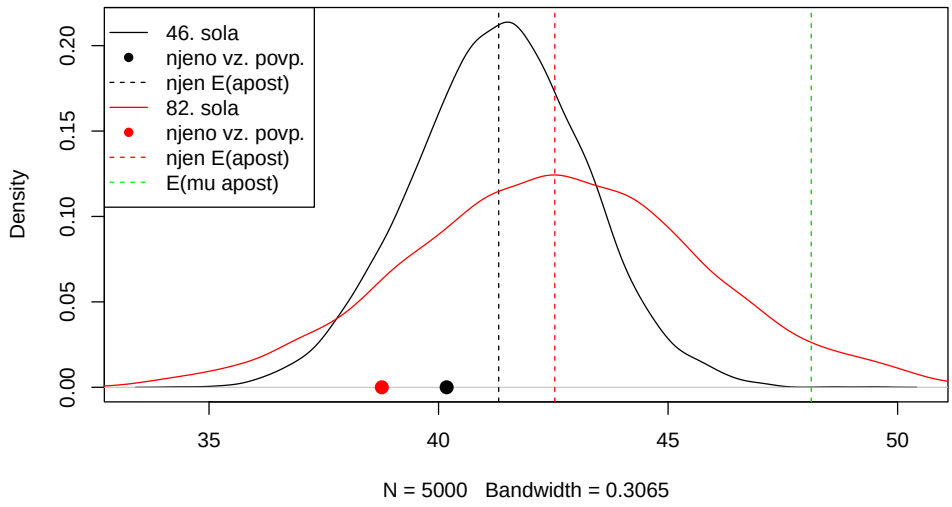
Mozne vrednosti aposteriorne morajo biti vsebovane v tistih od apriorne, ce:

- Zelimo imeti objektiven Bayesov pristop (tj. sibke apriorne, ki ne smejo informirati aposteriornih).
- Si slednje lahko privoscimo glede na stevilo enot oz. imamo dovolj podatkov, ki so zmozni informirati nase parametre (ce si ne moremo, je verjetno modeliranje parametra kljub odstopanjem aposteriorne od apriorne se zmeraj boljse kakor fiksiranje na neko vrednost, kar nekako ustreza apriorni z gostoto 1 v tej vrednosti).
- Ni namen modela drugacen, npr. skrciti koeficiente v regresiji proti nic.

3.3.1 *Shrinkage*

Oglejmo si aposteriorni porazdelitvi dveh sol:

```
plot(density(muGroups.all[,46]), type="l", main="")
points(pod.sole[46,]$povprecje, 0, pch=16, cex=1.5)
abline(v = mean(muGroups.all[,46]), lty=2)
lines(density(muGroups.all[,82]), type="l", col="red")
points(pod.sole[82,]$povprecje, 0, pch=16, cex=1.5, col="red")
abline(v = mean(muGroups.all[,82]), lty=2, col="red")
abline(v = mean(mu.all), lty=2, col="green3")
legend("topleft", c("46. sola", "njeno vz. povp.", "njen E(apost)",
                    "82. sola", "njeno vz. povp.", "njen E(apost)",
                    "E(mu apost)"),
       col=c("black","black","black","red","red","red","green"), lty=c(1,NA,2,1,NA,2,2),
       pch=c(NA,16,NA,NA,16,NA))
```



Navidez paradoksalno, saj je vzorcno povprecje 82. sole manjse od 46., medtem ko je pricakovana vrednosti aposteriorne porazdelitve velja ravno obratno.

Iz pogojne porazdelitve μ_j glede na vse ostalo vemo, da je:

$$\mathrm{E}(\mu_j \mid \text{vse ostalo}) = \frac{n_j/\sigma^2}{n_j/\sigma^2 + 1/\eta^2}\bar{x}_{\cdot j} + \frac{1/\eta^2}{n_j/\sigma^2 + 1/\eta^2}\mu$$

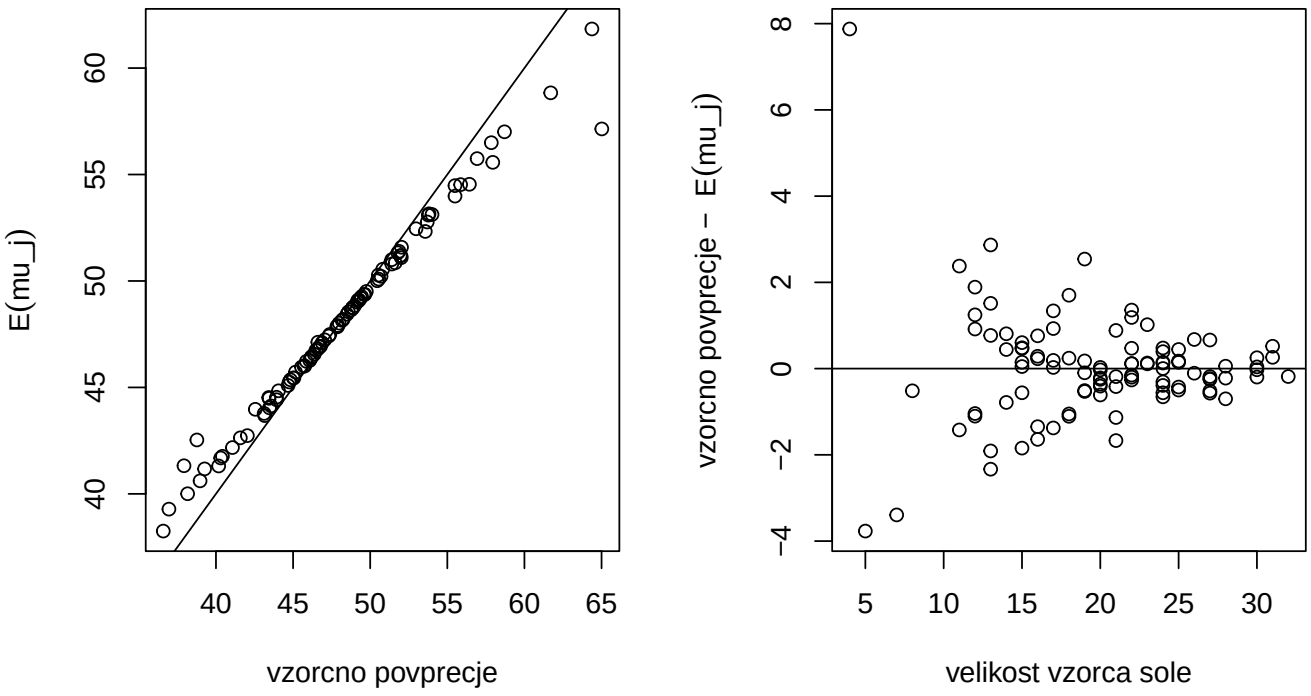
Pricakovana vrednost μ_j se torej pomakne proti skupnemu populacijskemu povprecju μ.

Ta efekt je pri velikem n_j majhen.

Poanta (zaradi katere zgornje ni paradoksalno): Pri soli z majhnim vzorcem smo lahko bolj zgresili povprecje te sole in zato bolj verjamemo skupnemu povprecju μ kot njenemu vzorcnemu povprecju oz. se k μ premaknemo bolj kot pri veliki soli.

Ocene povprecij sol (pricakovane vrednosti aposteriornih porazdelitev μ_j) se torej skrcijo (*shrinkage*) proti skupnemu povprecju μ, v primerjavi z vzorcnimi povprecji sol $\bar{x}_{\cdot j}$, kjer je ta premik vecji pri manjsi velikosti vzorca sole n_j. Slednje je jasno prikazano spodaj.

```
pod.sole$EmuGroups = colMeans(muGroups.all)

par(mfrow=c(1,2))
plot(pod.sole$povprecje, pod.sole$EmuGroups,
     xlab = "vzorcno povprecje", ylab = expression(E(mu_j)))
abline(a = 0, b = 1)

plot(pod.sole$n, pod.sole$povprecje - pod.sole$EmuGroups,
     xlab = "velikost vzorca sole",
     ylab = expression(paste("vzorcno povprecje - "," ",E(mu_j), sep="")))
abline(h = 0)
```



4 O ideji hierarhicnih modelov na splosno...

... vendar ob razmisljanju na nasem primeru.

Zamenljivost (exchangeability)

Znotraj modela smo predpostavili, da so tako meritve znotraj sol $X_{1,j}, \ldots, X_{n_j,j}$ zamenljive (tj. slucajni vzorec, torej standardna predpostavka), kakor tudi populacijska povprecja sol $\mu_1, \ldots, \mu_m$. Ali je to smiselno?

Nismo pa predpostavili, da so vse meritve $X_{1,1}, \ldots, X_{n_1,1}, \ldots, X_{m,1}, \ldots, X_{n_m,m}$ med seboj zamenljive. Zakaj?

Zamenjljivosti med vsemi podatki nismo predpostavili, saj smo imeli dodaten podatek za pripadnost šoli.

Ker ob tem nismo imeli nobenega dodatnega podatka o učencih (npr. izobrazba staršev, poklic staršev, ipd.), smo predpostavili zamenljivost znotraj šol - odsotnost informacije je torej implicirala zamenljivost :)

Kaj so prednosti/slabosti hierarhicnega modela v primerjavi z modelom z le enim skupnim povprecjem?

Primerjamo torej z modelom:

$$(X_{1,1}, \ldots, X_{n_1,1}, \ldots, X_{m,1}, \ldots, X_{n_m,m}) \mid \mu, \sigma^2 \sim \text{n.e.p. } N(\mu, \sigma^2),$$

tj. dvorazsezni normalni model, kjer podatke vseh sol zdruzimo v en vzorec.

Zanemarimo informacijo v podatkih, ki je lahko pomembna za izid.
Model ni dovolj fleksibilen.

Kaj so prednosti/slabosti hierarhicnega modela v primerjavi z modelom z vsemi razlicnimi nevezanimi povprecj?

Primerjamo torej z modelom:

$$(X_{1,j}, \ldots, X_{n_j,j}) \mid \mu_j, \sigma^2 \sim \text{n.e.p. } N(\mu_j, \sigma^2),$$

kjer nadalje velja

$$\begin{aligned} \pi(\mu_1, \ldots, \mu_m, \sigma^2) &= \pi(\mu_1, \ldots, \mu_m) \cdot \pi(\sigma^2) \\ &= \pi(\mu_1) \cdot \ldots \cdot \pi(\mu_m) \cdot \pi(\sigma^2). \end{aligned}$$

Ocenjujemo veliko parametrov, kar lahko vodi v preveliko prileganje modela podatkom (ang. overfitting).
Ker parametri niso vezani, potrebujemo dovolj velike vzorce za vsako šolo (pri vezanih parametrih si manjše šole lahko "izposodijo informacijo" od drugih).
Nimamo parametra, ki predstavlja povprečje celotne države.

7. sklop: Regresijski modeli

Tu predstavimo celotno generirano poročilo iz izvorne datoteke, vendar študentom ne priporočamo generiranje celotnega poročila iz izvorne datoteke, saj bo to trajalo dolgo časa (veliko modelov, veliko iteracij) - raje preglejte in prevedite izvorno kodo po delih.

1 Primer ocenjevanja povprecja

Na preteklih vajah smo za naslednji vzorec visin (metri) studentov moskega spola ocenjevali povprecje.

```
x <- c(1.91, 1.94, 1.68, 1.75, 1.81, 1.83, 1.91, 1.95, 1.77, 1.98,
       1.81, 1.75, 1.89, 1.89, 1.83, 1.89, 1.99, 1.65, 1.82, 1.65,
       1.73, 1.73, 1.88, 1.81, 1.84, 1.83, 1.84, 1.72, 1.91, 1.63)
```

To smo naredili na naslednje nacine:

1. Znana varianca, eksaktno s konjugirano apriorno porazdelitvijo, vzeli smo sibko informativno
2. Znana varianca, preko Metropolis-Hastings algoritma s konjugirano apriorno porazdelitvijo, vzeli smo sibko informativno (enaka kakor pri 1.)
3. Dvoparametricni model, eksaktno s konjugirano apriorno porazdelitvijo, vzeli smo sibko informativno
4. Dvoparametricni model, preko Metropolis-Hastings algoritma s sibko informativno apriorno porazdelitvijo

Kaj vse smo morali pri teh pristopih dolociti?

Sedaj povprecje ocenimo s pomocjo paketov v R.

Ocenjevanje povprecja je poseben primer linearne regresije, ki vsebuje le konstanten clen (brez drugih neodvisnih/pojasnjevalnih spremenljivk).

Standardne oznake v regresiji (izid y):

```
y <- x
```

1.1 BayesX

Bistvo: Spoznamo knjiznico BayesX.

```
#install.packages("R2BayesX")
library(R2BayesX)
```

```
## Loading required package: BayesXsrc
## Loading required package: colorspace
## Loading required package: mgcv
## Loading required package: nlme
## This is mgcv 1.8-42. For overview type 'help("mgcv-package")'.
```

```
bayesx.norm <- bayesx(y ~ 1, family = "gaussian", method = "MCMC")
```

Kaj vse smo morali dolociti? Potrebno dolociti le formulo, ostalo prednastavljeno (default).

Kaj lahko spremenimo?

Poglejte si pomoc za bayesx in bayesx.control (parametri iterations, burnin, step, hyp.prior).

Povzetek rezultatov:

```
summary(bayesx.norm)
```

```
## Call:
## bayesx(formula = y ~ 1, family = "gaussian", method = "MCMC")
##
## Fixed effects estimation results:
##
## Parametric coefficients:
##               Mean     Sd   2.5%    50%  97.5%
## (Intercept) 1.8208 0.0182 1.7841 1.8214 1.8552
##
## Scale estimate:
##          Mean     Sd   2.5%    50%  97.5%
## Sigma2 0.0103 0.0029 0.0062 0.0099 0.0175
##
## family = gaussian  method = MCMC  N = 30  iterations = 12000
## burnin = 2000  step = 10
```

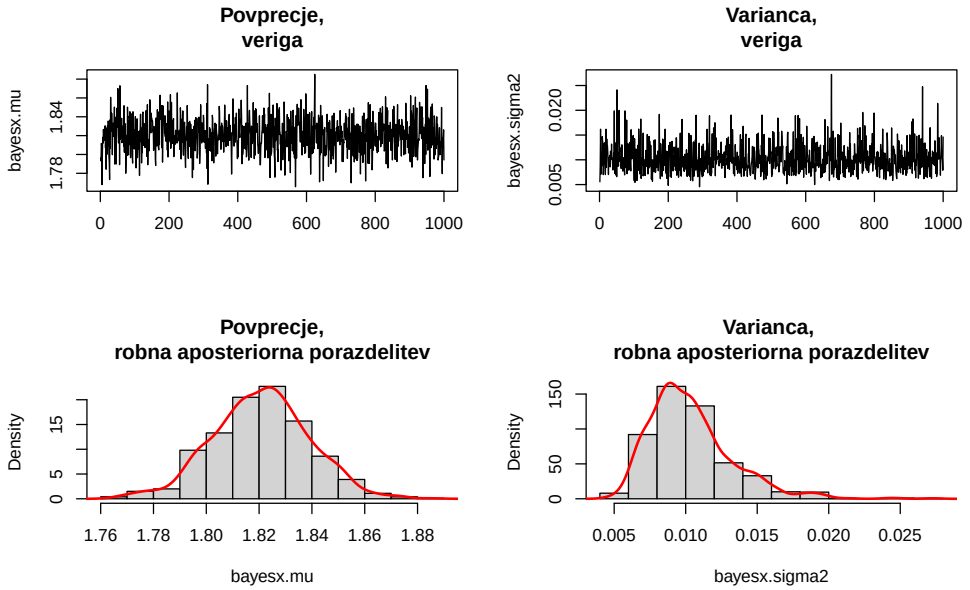
Takole dobimo vzorce:

```
bayesx.mu <- attr(bayesx.norm$fixed.effects, "sample")[,1]
bayesx.sigma2 <- attr(bayesx.norm$variance, "sample")
```

dolzina = (12000-2000)/10, tj. (iterations-burnin)/step

Narisemo vzorce:

```
par(mfrow = c(2, 2))
plot(bayesx.mu, type = "l", main = "Povprecje,\nveriga", xlab = "")
plot(bayesx.sigma2, type = "l", main = "Varianca,\nveriga", xlab = "")
hist(bayesx.mu, prob = T, main = "Povprecje,\nrobna aposteriorna porazdelitev")
lines(density(bayesx.mu), col = "red", lwd = 2)
hist(bayesx.sigma2, prob = T, main = "Varianca, \nrobna aposteriorna porazdelitev")
lines(density(bayesx.sigma2), col = "red", lwd = 2)
```



1.2 Nimble

Bistvo: Spoznamo knjiznico Nimble.

```
library(nimble)
```

```
## nimble version 1.0.1 is loaded.
## For more information on NIMBLE and a User Manual,
## please visit https://R-nimble.org.
##
## Note for advanced users who have written their own MCMC samplers:
##   As of version 0.13.0, NIMBLE's protocol for handling posterior
##   predictive nodes has changed in a way that could affect user-defined
##   samplers in some situations. Please see Section 15.5.1 of the User Manual.
##
## Attaching package: 'nimble'
## The following object is masked from 'package:stats':
##
##     simulate
```

Stirje osnovni gradniki:

```
# 1. Dolocimo model in apriorne porazdelitve parametrov:
code <- nimbleCode({
  mu ~ dnorm(0, 0.00001) #default za drugi parameter je precision
  sigma2 ~ dunif(0, 10000)
  for (i in 1:n) {
    y[i] ~ dnorm(mu, var = sigma2)
  }
})

code #nic ne naredi, le shrani si predstavitev modela
```

```
## {
##     mu ~ dnorm(0, 1e-05)
##     sigma2 ~ dunif(0, 10000)
##     for (i in 1:n) {
##         y[i] ~ dnorm(mu, var = sigma2)
##     }
## }
```

```
# 2. Dolocimo konstante v modelu:
constants <- list(n = length(y))
```

```
# 3. Dolocimo podatke:
data <- list(y = y)
```

```
# 4. Dolocimo zacetne vrednosti za verige parametrov:
inits <- list(mu = 0,
              sigma2 = 1)
```

```
Rmodel <- nimbleModel(code, constants, data, inits) #le model zgradis
```

```
## Defining model
```

```
## Building model
```

```
## Setting data and initial values
```

```
## Running calculate on model
##   [Note] Any error reports that follow may simply reflect missing values in model var
```

```
## Checking model sizes and dimensions
```

```
Rmodel$initializeInfo() #preverimo, da je use ok
```

```
##   [Note] All model variables are initialized.
```

```
conf <- configureMCMC(Rmodel) #zacetek inference
```

```
## ===== Monitors =====
```

```
## thin = 1: mu, sigma2
## ===== Samplers =====
## RW sampler (1)
##   - sigma2
## conjugate sampler (1)
##   - mu
```

```
Rmcmc <- buildMCMC(conf)
```

S spodnjim pozenemo verigo, vendar v R-u in zato je zelo pocasno (spodaj le 100 iteracij)! Ne uporabljati tega!!!

```
samples <- runMCMC(Rmcmc, 100)
```

```
##   [Warning] Running an uncompiled MCMC algorithm, use compileNimble() for faster exec
```

```
## running chain 1...
```

```
## |-------------|-------------|-------------|-------------|
## |-------------------------------------------------------|
```

Do sedaj smo delali v R. Prevedemo v C++:

```
Cmodel <- compileNimble(Rmodel)
```

```
## Compiling
##   [Note] This may take a minute.
##   [Note] Use 'showCompilerOutput = TRUE' to see C++ compilation details.
```

```
Cmcmc <- compileNimble(Rmcmc, project = Cmodel)
```

```
## Compiling
##   [Note] This may take a minute.
##   [Note] Use 'showCompilerOutput = TRUE' to see C++ compilation details.
```

```
samples <- runMCMC(Cmcmc, 100)   #ekspres
```

```
## running chain 1...
```

```
## |-------------|-------------|-------------|-------------|
## |-------------------------------------------------------|
```

```
samples <- runMCMC(Cmcmc, 1000)  #ekspres
```

```
## running chain 1...
```

```
## |-------------|-------------|-------------|-------------|
## |-------------------------------------------------------|
```

```
samples <- runMCMC(Cmcmc, 10000) #ekspres
```

```
## running chain 1...
```

```
## |-------------|-------------|-------------|-------------|
## |-------------------------------------------------------|
```

Bliznjica od stirih osnovnih gradnikov do koncnih rezultatov (brez vmesnih korakov):

```
samples <- nimbleMCMC(code, constants, data, inits, niter=10000)
```

```
## Defining model
## Building model
## Setting data and initial values
## Running calculate on model
##   [Note] Any error reports that follow may simply reflect missing values in model var
## Checking model sizes and dimensions
## Checking model calculations
## Compiling
##   [Note] This may take a minute.
##   [Note] Use 'showCompilerOutput = TRUE' to see C++ compilation details.
## running chain 1...
## |-------------|-------------|-------------|-------------|
## |-------------------------------------------------------|
```

Oglejmo si rezultate:

```
dim(samples)
```

```
## [1] 10000     2
```

```
head(samples)
```

```
##            mu    sigma2
## [1,] 1.958654 1.0000000
## [2,] 1.601149 1.0000000
## [3,] 2.118685 1.0000000
## [4,] 1.795769 0.9768545
## [5,] 1.627561 0.9768545
## [6,] 1.856253 0.2467002
```

```
samplesSummary(samples) #funkcija za povzetek rezultatov
```

```
##             Mean     Median    St.Dev.  95%CI_low  95%CI_upp
## mu     1.8203116 1.82035755 0.01997874 1.78145946 1.85824032
## sigma2 0.0117338 0.01071065 0.02226094 0.00662094 0.01900782
```

```
summary(bayesx.norm) #primerjava z BayesX - seveda zelo podobno
```

```
## Call:
## bayesx(formula = y ~ 1, family = "gaussian", method = "MCMC")
##
## Fixed effects estimation results:
##
## Parametric coefficients:
##               Mean     Sd   2.5%    50%  97.5%
## (Intercept) 1.8208 0.0182 1.7841 1.8214 1.8552
##
## Scale estimate:
##          Mean     Sd   2.5%    50%  97.5%
## Sigma2 0.0103 0.0029 0.0062 0.0099 0.0175
##
## family = gaussian  method = MCMC  N = 30  iterations = 12000
## burnin = 2000  step = 10
```

```
mean(y) #vzorcno povprecje
```

```
## [1] 1.820667
```

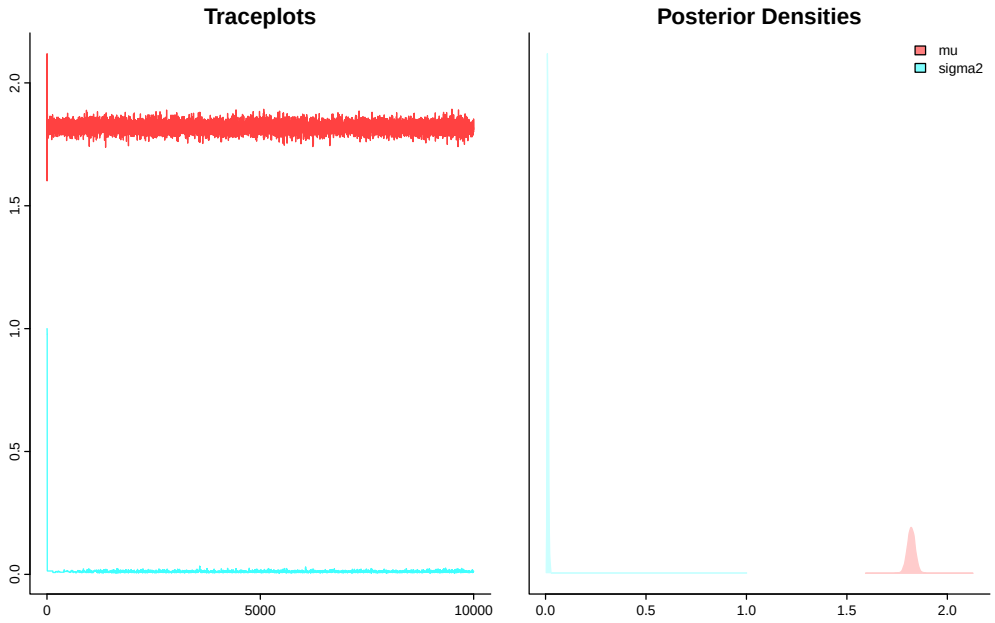
```
var(y)  #vzorcna varianca
```

```
## [1] 0.009716782
```

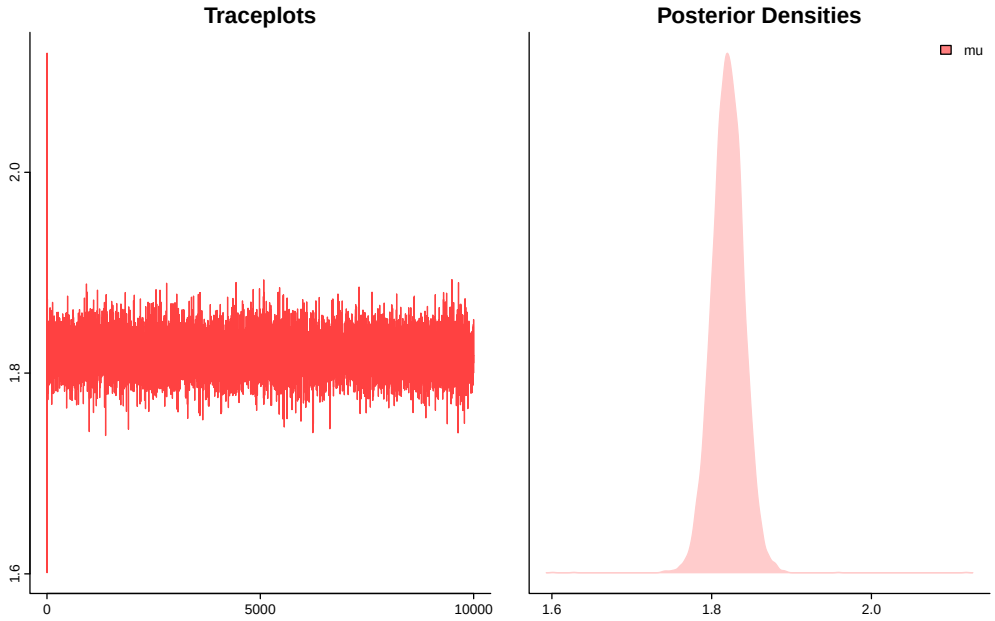
```
#BTW: y je bil generiran iz N(mean=1.85, sigma2=0.01)
#Oba prava (populacijska) parametra sta vsebovana v credible interval.
```

Graficno predstavimo rezultate:

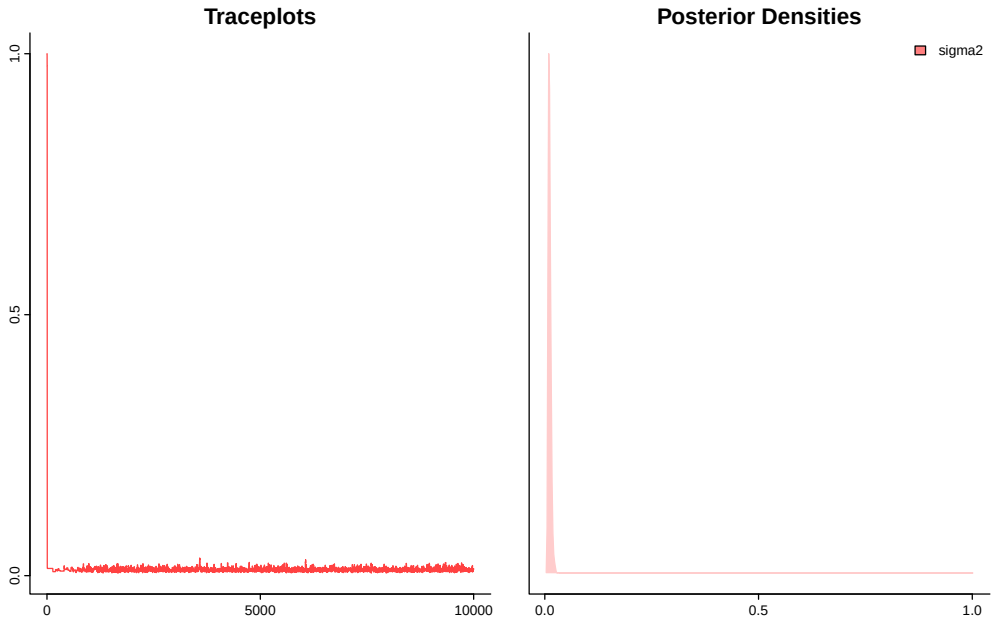
```
library(basicMCMCplots)
samplesPlot(samples)
```




```
samplesPlot(samples, var = "mu")
```




```
samplesPlot(samples, var = "sigma2")
```



2 (Multipla) linearna regresija

Podane imamo podatke o slovenskih ekipnih sportnicah (kosarkasice, odbojkasice in rokometasice), stare od 10 do 30 let. Preucevali so poskodbe kolen.

Podatki so bili podlaga za clanek R. Vauhnik et al., *Rate and risk of anterior cruciate ligament injury among sportswomen in Slovenia* (2011).

```
#sportnice = read.delim("sportnice.txt")
#sportnice = sportnice[sportnice$starost<30,]
#dump("sportnice", file="sportnice.R")

source("sportnice.R")
str(sportnice)
```

```
## 'data.frame':    660 obs. of  16 variables:
##  $ pacient_: Factor w/ 669 levels "2071287        ",..: 485 182 368 27 314 535 347 62
##  $ starost : int  15 18 21 19 21 23 19 19 15 16 ...
##  $ teza    : num  55 58 62 68 60 63 61 65 75 55.5 ...
##  $ visina  : num  170 168 172 175 164 ...
##  $ bmi     : num  16.2 17.3 18 19.4 18.3 ...
##  $ sport   : Factor w/ 3 levels "kosarka","odbojka",..: 2 2 2 2 2 2 2 2 2 2 ...
##  $ trening : num  8 1.5 7.5 8 7.5 7.5 8 8 8 8 ...
##  $ kontrace: Factor w/ 2 levels "da","ne": 2 1 1 1 2 1 1 2 2 2 ...
##  $ ndt_righ: num  0.5 0.5 2 0.5 0.7 0.6 0.8 0.2 0.5 0.5 ...
##  $ kextr   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ lb30r   : num  7.5 6 5 7 10.3 ...
##  $ ci_right: num  5.5 4 3 4 6.33 4 3.5 2.33 5 3 ...
```

```
##  $ ndt_left: num  0.5 0.5 2 0.5 0.5 0.3 0.3 0.2 0.2 0.5 ...
##  $ kextl   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ lb30l   : num  6.5 6 5 7 9 5 4.5 5 6 5 ...
##  $ ci_left : num  4.5 4 3 5 5 3 3 3 4 3 ...
```

2.1 Ali je kolicina treninga povezana z jemanjem kontracepcijskih tablet?

Preverimo najprej s frekventisticnim pristopom:

```
fit.lm <- lm(trening ~ kontrace, data = sportnice)
summary(fit.lm)
```

```
##
## Call:
## lm(formula = trening ~ kontrace, data = sportnice)
##
## Residuals:
##    Min     1Q Median     3Q    Max
##  -6.39  -1.89  -0.39   2.11  12.11
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   7.8900     0.2335  33.785   <2e-16 ***
## kontracene   -0.4792     0.2657  -1.804   0.0717 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.86 on 658 degrees of freedom
## Multiple R-squared:  0.00492,    Adjusted R-squared:  0.003408
## F-statistic: 3.254 on 1 and 658 DF,  p-value: 0.07172
```

Komentirajte rezultate. Ali vas presenecajo? Kako bi jih lahko pojasnili?

```
fit2.lm <- lm(trening ~ kontrace + starost, data = sportnice)
summary(fit2.lm)
```

```
##
## Call:
## lm(formula = trening ~ kontrace + starost, data = sportnice)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7.4729 -1.8956 -0.3956  1.4636 11.6608
##
## Coefficients:
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  3.69232    0.74069   4.985 7.94e-07 ***
## kontracene   0.03492    0.27299   0.128    0.898
## starost      0.21122    0.03547   5.956 4.22e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.788 on 657 degrees of freedom
## Multiple R-squared:  0.05589,    Adjusted R-squared:  0.05302
## F-statistic: 19.45 on 2 and 657 DF,  p-value: 6.238e-09
```

Bistvo:

- Specifikacija linearnega modela s paketom nimble.
- Lastnosti linearne regresije v splosnem (ni specificno za Bayesovo analizo) - kljucna razlika med enostavno in multiplo regresijo. Ker sta tako trening kot jemanje kontracepcijskih tablet povezana s starostjo (razpon 11-29 let), lahko pravi efekt jemanja kontracepcije opazimo sele, ko sportnice izenacimo glede na starost, tj. v model dodamo neodvisno spremenljivko starost.

Bayesovski pristop s paketom BayesX:

```
fit.bayesx <- bayesx(trening ~ kontrace, data = sportnice, family = "gaussian", method =
```

```
## Note: created new output directory 'C:/Users/ninar/AppData/Local/Temp/Rtmpclf3Ph/baye
```

```
summary(fit.bayesx)
```

```
## Call:
## bayesx(formula = trening ~ kontrace, data = sportnice, family = "gaussian",
##     method = "MCMC")
##
## Fixed effects estimation results:
##
## Parametric coefficients:
##                Mean      Sd    2.5%     50%  97.5%
## (Intercept)  7.8947  0.2344  7.4166  7.8930 8.3366
## kontracene  -0.4871  0.2666 -1.0244 -0.4917 0.0712
##
## Scale estimate:
##          Mean     Sd   2.5%    50%  97.5%
## Sigma2 8.2270 0.4605 7.3690 8.2020 9.2036
##
## N = 660  burnin = 2000  method = MCMC  family = gaussian
## iterations = 12000  step = 10
```

```
fit2.bayesx <- bayesx(trening ~ kontrace + starost, data = sportnice, family = "gaussian
```

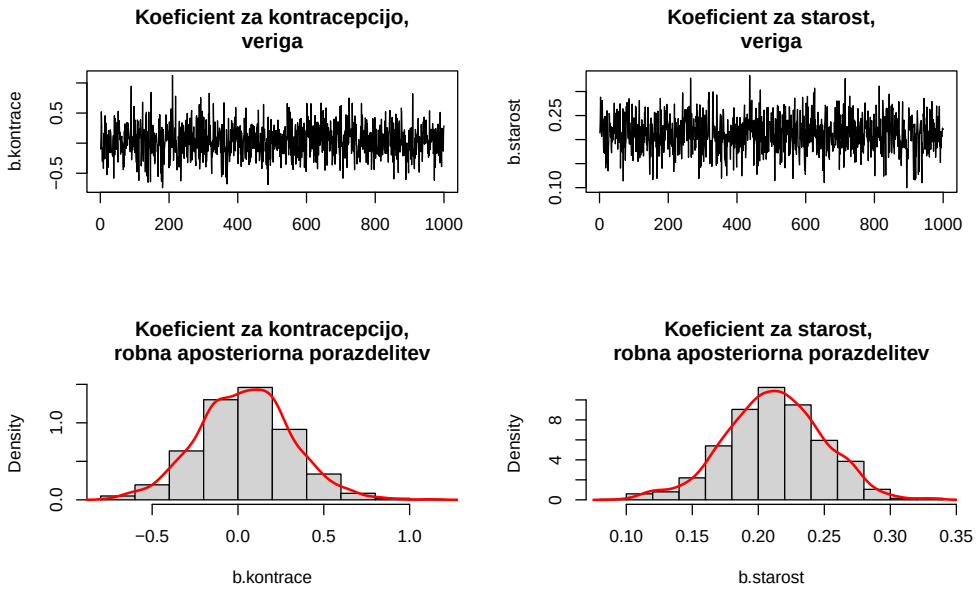
```
## Note: created new output directory 'C:/Users/ninar/AppData/Local/Temp/Rtmpclf3Ph/baye
```

```
summary(fit2.bayesx)
```

```
## Call:
## bayesx(formula = trening ~ kontrace + starost, data = sportnice,
##     family = "gaussian", method = "MCMC")
##
## Fixed effects estimation results:
##
## Parametric coefficients:
##                Mean      Sd    2.5%     50%  97.5%
## (Intercept)  3.6687  0.7608  2.2200  3.6733 5.2632
## kontracene   0.0395  0.2691 -0.4852  0.0421 0.5890
## starost      0.2122  0.0367  0.1353  0.2125 0.2809
##
## Scale estimate:
##          Mean     Sd   2.5%    50%  97.5%
## Sigma2 7.8006 0.4417 7.0000 7.7846 8.6815
##
## N = 660  burnin = 2000  method = MCMC  family = gaussian
## iterations = 12000  step = 10
```

```
b.kontrace <- attr(fit2.bayesx$fixed.effects, "sample")[,2]
b.starost <- attr(fit2.bayesx$fixed.effects, "sample")[,3]
```

```
par(mfrow = c(2, 2))
plot(b.kontrace, type = "l", main = "Koeficient za kontracepcijo,\nveriga",
     xlab = "")
plot(b.starost, type = "l", main = "Koeficient za starost,\nveriga",
     xlab = "")
hist(b.kontrace, prob = T, main = "Koeficient za kontracepcijo,\nrobna aposteriorna pora
lines(density(b.kontrace), col = "red", lwd = 2)
hist(b.starost, prob = T, main = "Koeficient za starost, \nrobna aposteriorna porazdelit
lines(density(b.starost), col = "red", lwd = 2)
```



Naloga: Uporabite nimble, kjer za apriorne porazdelitve regresijskih koeficientov in konstante vzemite dnorm(0, sd = 100) in za sigma dunif(0, 100). Oglejte si rezultate in jih graficno predstavite.

```
code <- nimbleCode({
  beta0 ~ dnorm(0, sd = 100) #tipicno vzamemo taksen neinformativen prior
  beta1 ~ dnorm(0, sd = 100)
  beta2 ~ dnorm(0, sd = 100)
  sigma ~ dunif(0, 100)
  for(i in 1:n) {
    y[i] ~ dnorm(beta0 + beta1*x1[i] + beta2*x2[i], sd = sigma)
  }
})

constants <- list(n = length(sportnice$trening),
                  x1 = sportnice$kontrace,
                  x2 = sportnice$starost)

data <- list(y = sportnice$trening)

inits <- list(beta0 = mean(sportnice$trening),
              beta1 = 0,
              beta2 = 0,
              sigma = 1)
```

```
Rmodel <- nimbleModel(code, constants, data, inits)

## Defining model
```

```
## Building model

## Setting data and initial values

## Running calculate on model
##   [Note] Any error reports that follow may simply reflect missing values in model var

## Checking model sizes and dimensions
```

```
conf <- configureMCMC(Rmodel)
```

```
## ===== Monitors =====
## thin = 1: beta0, beta1, beta2, sigma
## ===== Samplers =====
## RW sampler (1)
##   - sigma
## conjugate sampler (3)
##   - beta0
##   - beta1
##   - beta2
```

```
Rmcmc <- buildMCMC(conf)
Cmodel <- compileNimble(Rmodel)
```

```
## Compiling
##   [Note] This may take a minute.
##   [Note] Use 'showCompilerOutput = TRUE' to see C++ compilation details.
```

```
Cmcmc <- compileNimble(Rmcmc, project = Cmodel)
```

```
## Compiling
##   [Note] This may take a minute.
##   [Note] Use 'showCompilerOutput = TRUE' to see C++ compilation details.
```

```
samples <- runMCMC(Cmcmc, niter = 1000)
```

```
## running chain 1...
```

```
## |-------------|-------------|-------------|-------------|
## |-------------------------------------------------------|
```

```
samples <- runMCMC(Cmcmc, niter = 12000, nburnin = 2000)
```

```
## running chain 1...
```

```
## |-------------|-------------|-------------|-------------|
## |-------------------------------------------------------|
```

```
dim(samples)
```

```
## [1] 10000     4
```

```
head(samples)
```

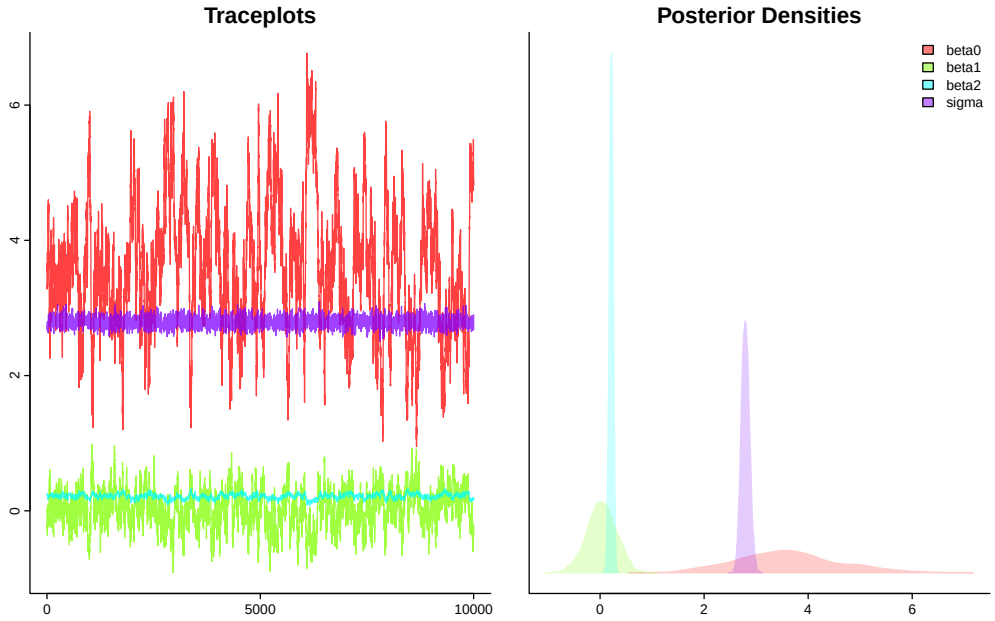
```
##          beta0        beta1     beta2    sigma
## [1,] 3.280025  0.003455009 0.2342466 2.671213
## [2,] 3.334296  0.022605490 0.2347469 2.671213
## [3,] 3.327467 -0.065825757 0.2442280 2.671213
## [4,] 3.284618 -0.180792257 0.2496379 2.671213
## [5,] 3.373578 -0.202982691 0.2405619 2.671213
## [6,] 3.641883 -0.231381431 0.2420262 2.671213
```

```
samplesSummary(samples)
```

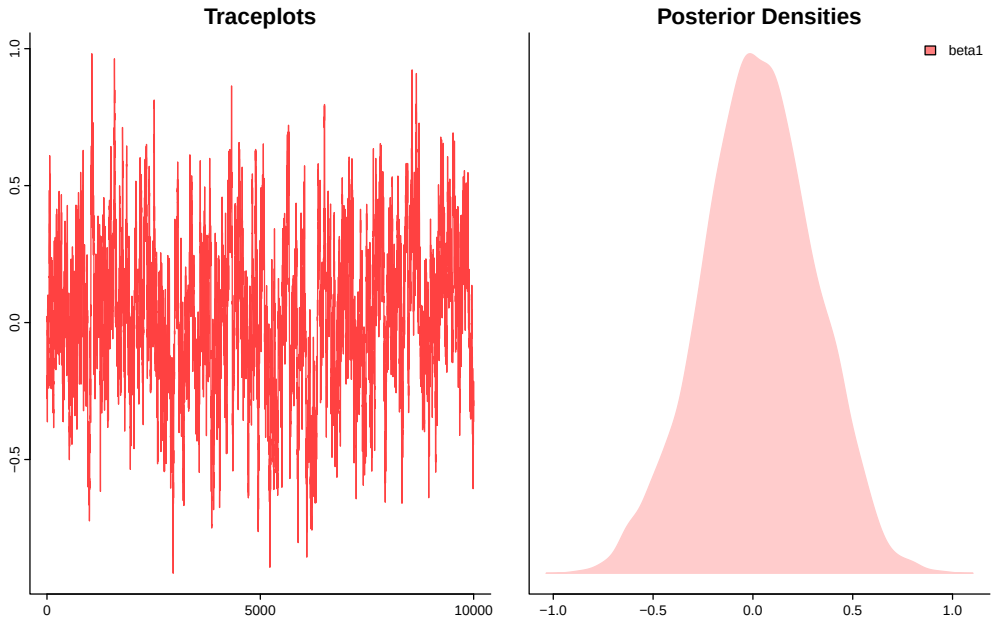
```
##              Mean     Median    St.Dev.  95%CI_low 95%CI_upp
## beta0 3.60666937 3.56373516 0.96421684  1.8348186 5.6311202
## beta1 0.02960792 0.02780825 0.28066319 -0.5361944 0.5638976
## beta2 0.21456301 0.21558832 0.03743681  0.1377039 0.2837378
## sigma 2.79384803 2.79212774 0.07748445  2.6478977 2.9514651
```

Graficno predstavimo rezultate:

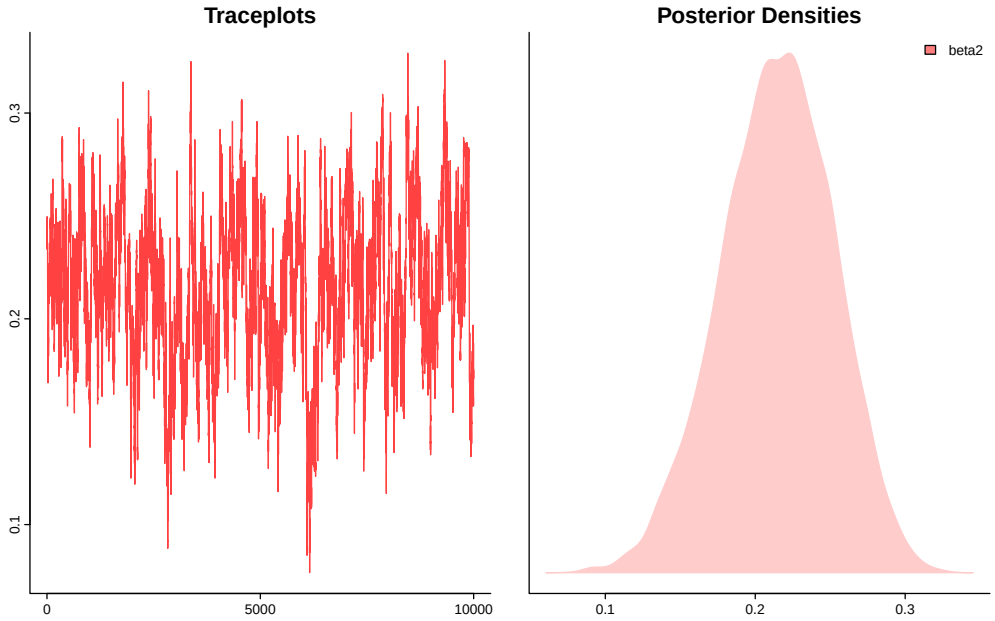
```
samplesPlot(samples)
```




```
samplesPlot(samples, var = "beta1")
```




```
samplesPlot(samples, var = "beta2")
```



Primerjamo z BayesX:

```
summary(fit2.bayesx)
```

```
## Call:
## bayesx(formula = trening ~ kontrace + starost, data = sportnice,
##     family = "gaussian", method = "MCMC")
##
## Fixed effects estimation results:
##
```

```
## Parametric coefficients:
##                Mean      Sd    2.5%     50%  97.5%
## (Intercept)  3.6687  0.7608  2.2200  3.6733 5.2632
## kontracene   0.0395  0.2691 -0.4852  0.0421 0.5890
## starost      0.2122  0.0367  0.1353  0.2125 0.2809
##
## Scale estimate:
##          Mean     Sd   2.5%    50%  97.5%
## Sigma2 7.8006 0.4417 7.0000 7.7846 8.6815
##
## N = 660  burnin = 2000  method = MCMC  family = gaussian
## iterations = 12000  step = 10
```

2.2 Isti primer z bolj ucinkovito specifikacijo modela in z generiranjem vec verig

Bistvo:

- Bolj ucinkovita specifikacija linearnega modela s paketom nimble.
- Kako preprosto generiramo vec verig in graficno preucimo konvergenco.

Bayesovski pristop s paketom nimble:

```
code <- nimbleCode({
  beta0 ~ dnorm(0, sd = 100)
  for(k in 1:p) { #lahko uporabimo for zanko za definicijo vseh apriornih porazdelitev
    beta[k] ~ dnorm(0, sd = 100)
  }
  sigma ~ dunif(0, 100)
  for(i in 1:n) {
    y[i] ~ dnorm(beta0 + inprod(beta[1:p], x[i, 1:p]), sd = sigma) #uporabimo funkcijo
  }
})

sportnice$kontrace = ifelse(sportnice$kontrace=="da", 1, 0)
X <- subset(sportnice, select = c("kontrace",
                                  "starost"))
p <- ncol(X)

constants <- list(n = length(sportnice$trening),
                  p = p,
                  x = X)

data <- list(y = sportnice$trening)

inits <- list(beta0 = mean(sportnice$trening),
```

```
              beta = rep(0, p),
              sigma = 1)
```

```
Rmodel <- nimbleModel(code, constants, data, inits)
```

```
## Defining model
```

```
## Building model
```

```
## Setting data and initial values
```

```
## Running calculate on model
##   [Note] Any error reports that follow may simply reflect missing values in model var
```

```
## Checking model sizes and dimensions
```

```
conf <- configureMCMC(Rmodel)
```

```
## ===== Monitors =====
## thin = 1: beta, beta0, sigma
## ===== Samplers =====
## RW sampler (1)
##   - sigma
## conjugate sampler (3)
##   - beta0
##   - beta[]  (2 elements)
```

```
Rmcmc <- buildMCMC(conf)
Cmodel <- compileNimble(Rmodel)
```

```
## Compiling
##   [Note] This may take a minute.
##   [Note] Use 'showCompilerOutput = TRUE' to see C++ compilation details.
```

```
Cmcmc <- compileNimble(Rmcmc, project = Cmodel)
```

```
## Compiling
##   [Note] This may take a minute.
##   [Note] Use 'showCompilerOutput = TRUE' to see C++ compilation details.
```

```
samples <- runMCMC(Cmcmc, niter = 1000)
```

```
## running chain 1...
```

```
## |-------------|-------------|-------------|-------------|
## |-------------------------------------------------------|
```

```
samples <- runMCMC(Cmcmc, niter = 12000, nburnin = 2000)
```

```
## running chain 1...
```

```
## |-------------|-------------|-------------|-------------|
## |-------------------------------------------------------|
```

```
dim(samples)
```

```
## [1] 10000     4
```

```
head(samples)
```

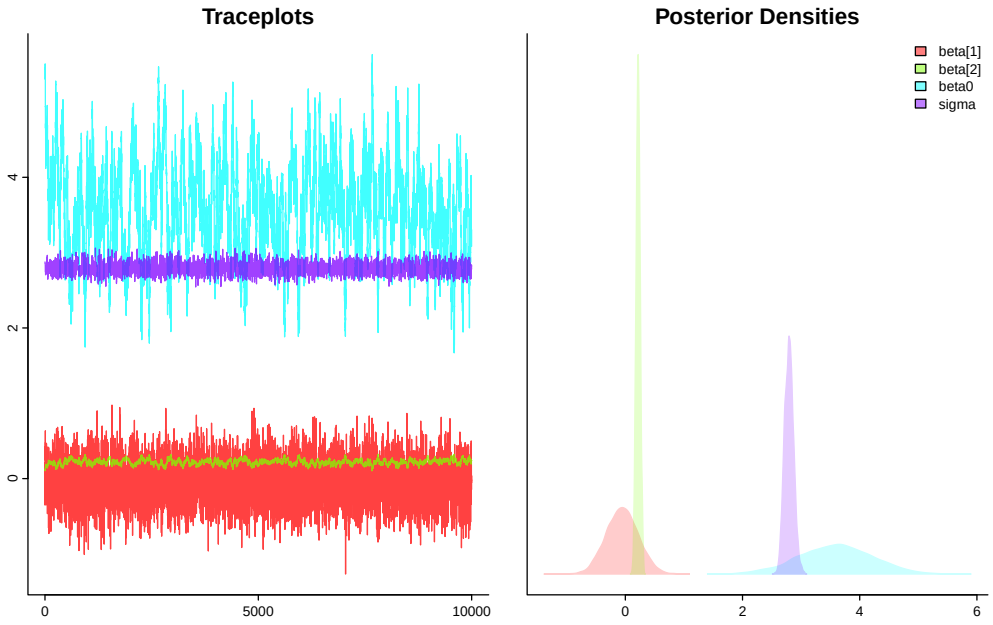
```
##          beta[1]   beta[2]    beta0    sigma
## [1,] -0.05066592 0.1253619 5.303779 2.876249
## [2,] -0.35190605 0.1192596 5.350620 2.856288
## [3,]  0.15670944 0.1049496 5.455203 2.856288
## [4,]  0.21429197 0.1077007 5.440144 2.856288
## [5,]  0.06461222 0.1095860 5.506047 2.856288
## [6,] -0.05880011 0.1179085 5.461835 2.856288
```

```
samplesSummary(samples)
```

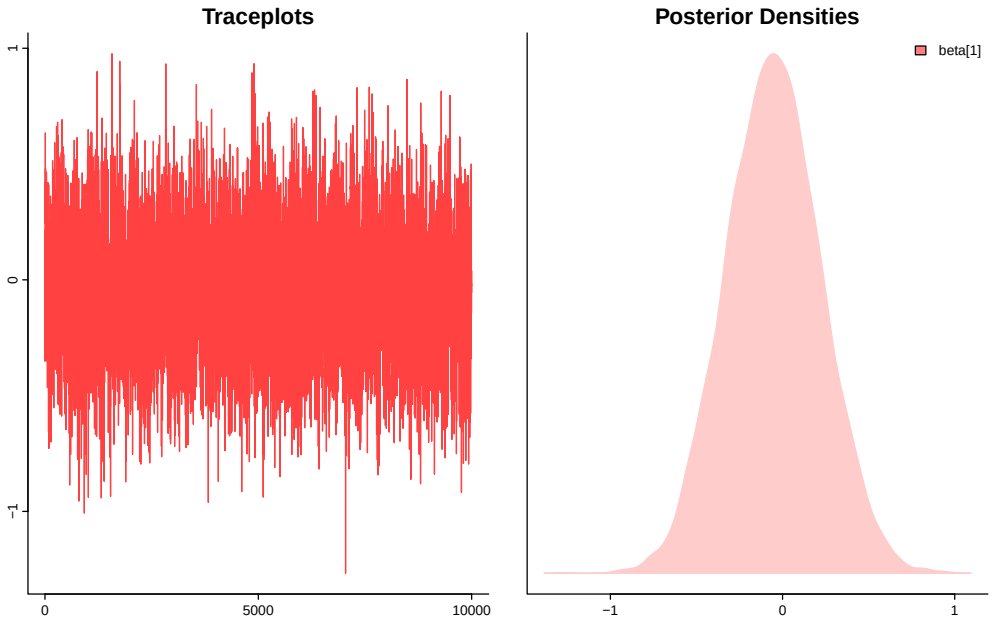
```
##                Mean      Median    St.Dev.  95%CI_low 95%CI_upp
## beta[1] -0.05189706 -0.05288237 0.27393993 -0.5764737 0.4839261
## beta[2]  0.21898917  0.21895438 0.03504209  0.1513766 0.2898851
## beta0    3.59040371  3.59879998 0.62281456  2.3377187 4.7996620
## sigma    2.79405128  2.79363184 0.07722947  2.6494137 2.9494305
```

Graficno predstavimo rezultate:

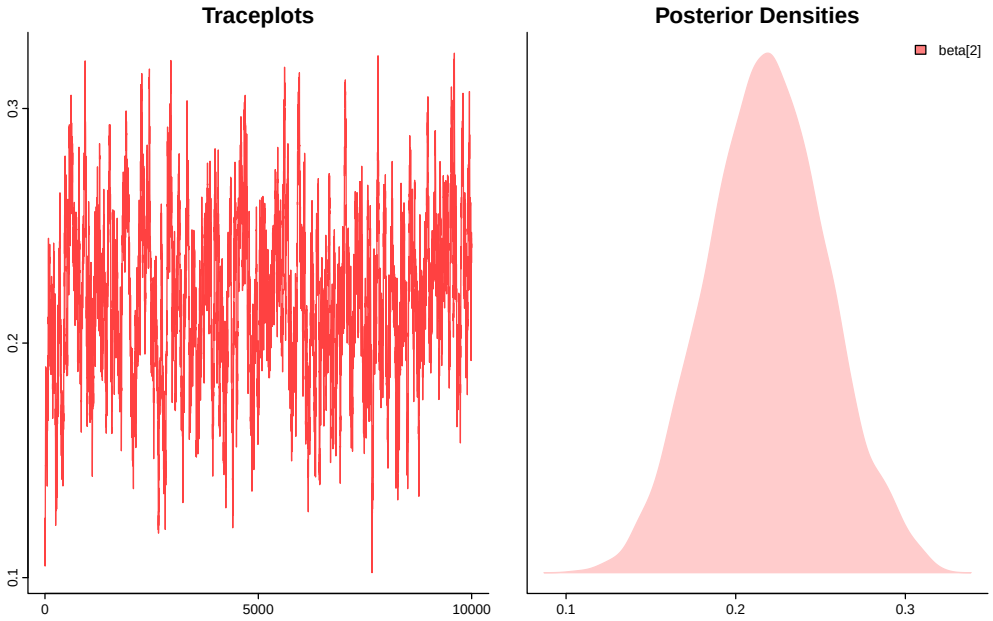
```
samplesPlot(samples)
```




```
samplesPlot(samples, var = "beta[1]")
```




```
samplesPlot(samples, var = "beta[2]")
```



Vec verig z nakljucnimi zacetnimi vrednostmi:

```
initsFunction <- function(){
  list(beta0 = rnorm(1),
       beta = rnorm(2),
       sigma = runif(1, min = 0, max = 10))
}

initsFunction()
```

```
## $beta0
## [1] -0.7165031
##
## $beta
## [1] -0.9493640 -0.6608608
##
## $sigma
## [1] 0.801283
```

```
# Z dodatnim parametrom v runMCMC dolocimo stevilo verig:
samplesList <- runMCMC(Cmcmc, niter = 12000, nburnin = 0,
                       nchains = 3, inits = initsFunction)
```

```
## running chain 1...
## |-------------|-------------|-------------|-------------|
## |-------------------------------------------------------|
## running chain 2...
## |-------------|-------------|-------------|-------------|
## |-------------------------------------------------------|
## running chain 3...
## |-------------|-------------|-------------|-------------|
## |-------------------------------------------------------|
```

```
str(samplesList)
```

```
## List of 3
##  $ chain1: num [1:12000, 1:4] 0.3884 0.0769 -0.1404 -0.1585 0.2912 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
##   .. ..$ : chr [1:4] "beta[1]" "beta[2]" "beta0" "sigma"
##  $ chain2: num [1:12000, 1:4] 1.92 3.33 3.84 3.96 3.57 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
##   .. ..$ : chr [1:4] "beta[1]" "beta[2]" "beta0" "sigma"
##  $ chain3: num [1:12000, 1:4] 2.56 2.97 2.98 2.54 2.52 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
##   .. ..$ : chr [1:4] "beta[1]" "beta[2]" "beta0" "sigma"
```

```
str(samplesList$chain1) #dobimo eno verigo
```

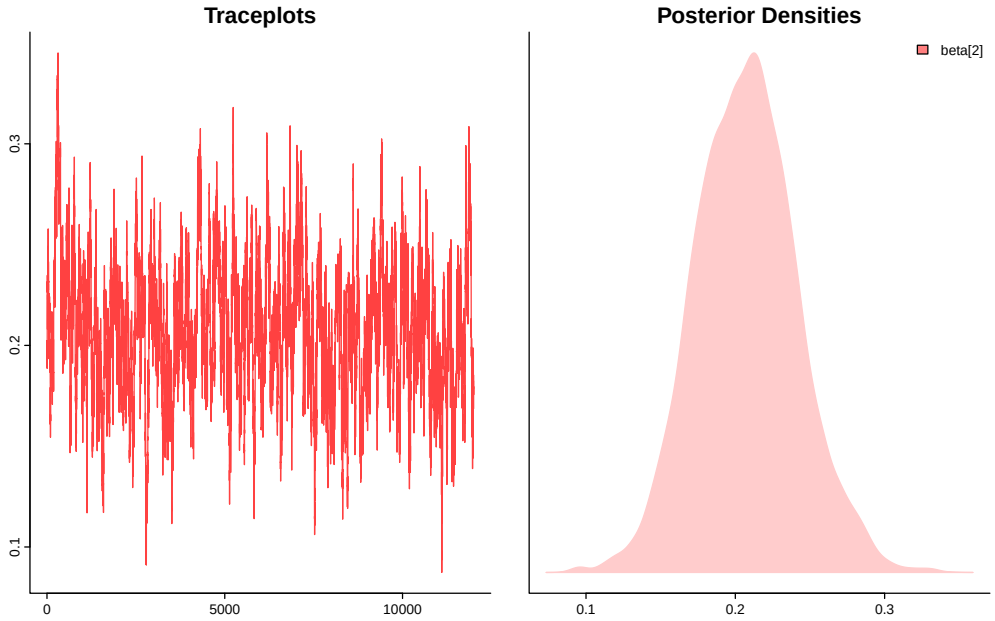
```
##  num [1:12000, 1:4] 0.3884 0.0769 -0.1404 -0.1585 0.2912 ...
##  - attr(*, "dimnames")=List of 2
##   ..$ : NULL
```

```
##   ..$ : chr [1:4] "beta[1]" "beta[2]" "beta0" "sigma"
```

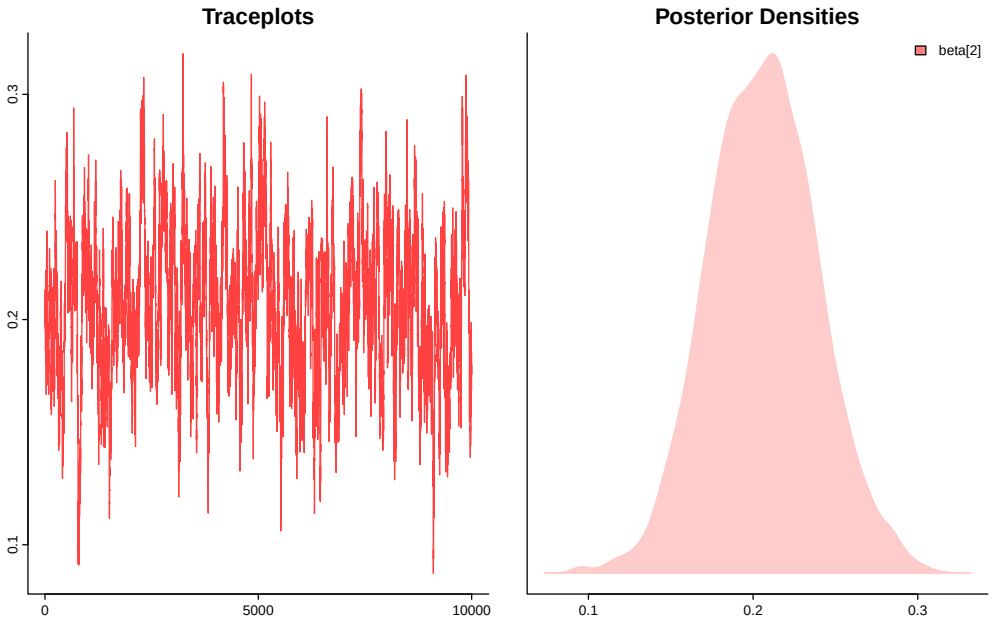
```
str(samplesList[[1]])   #ekvivalentno
```

```
##  num [1:12000, 1:4] 0.3884 0.0769 -0.1404 -0.1585 0.2912 ...
##  - attr(*, "dimnames")=List of 2
##   ..$ : NULL
##   ..$ : chr [1:4] "beta[1]" "beta[2]" "beta0" "sigma"
```

```
samplesPlot(samplesList[[1]], var = "beta[2]") #si narisemo 1 parameter iz 1 verige
```

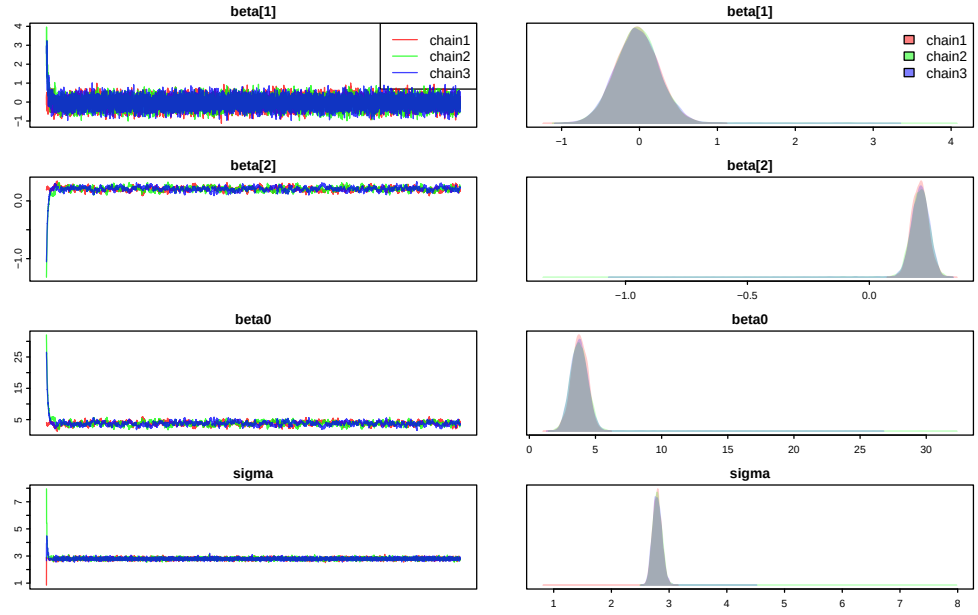



```
# Namesto, da bi ze zgoraj dolocili burnin, ga lahko tule:
samplesPlot(samplesList[[1]], var = "beta[2]", burnin = 2000)
```

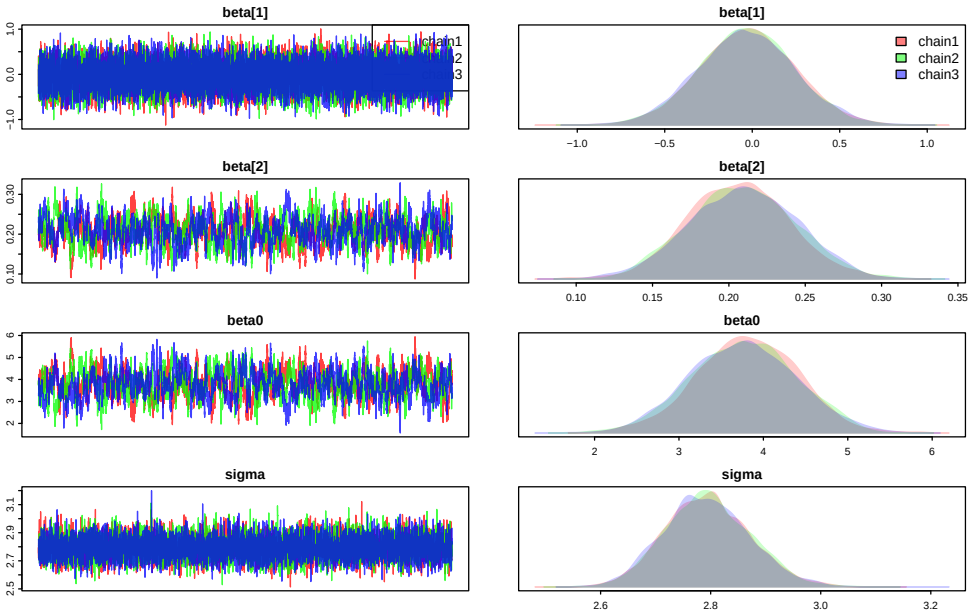


Ukaza za graficna prikaza vecih verig:

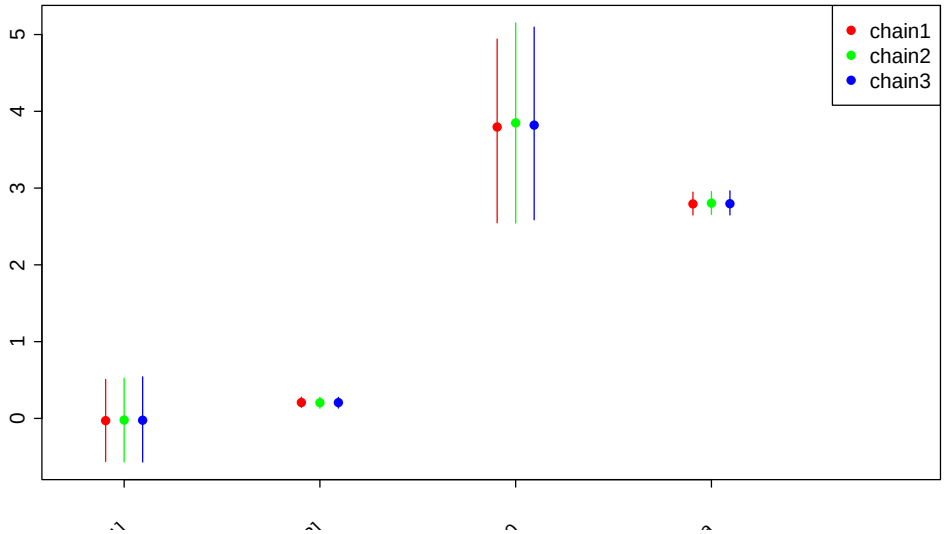
```
chainsPlot(samplesList) #dobimo trace plots in gostote aposteriornih porazdelitev
```



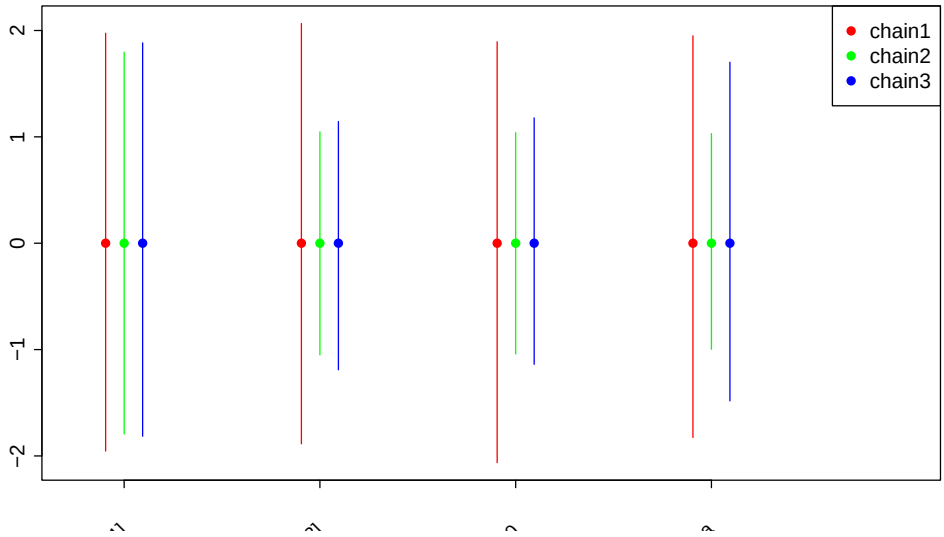

```
# Preprosto vzamemo nek burnin:
chainsPlot(samplesList, burnin = 2000)
```



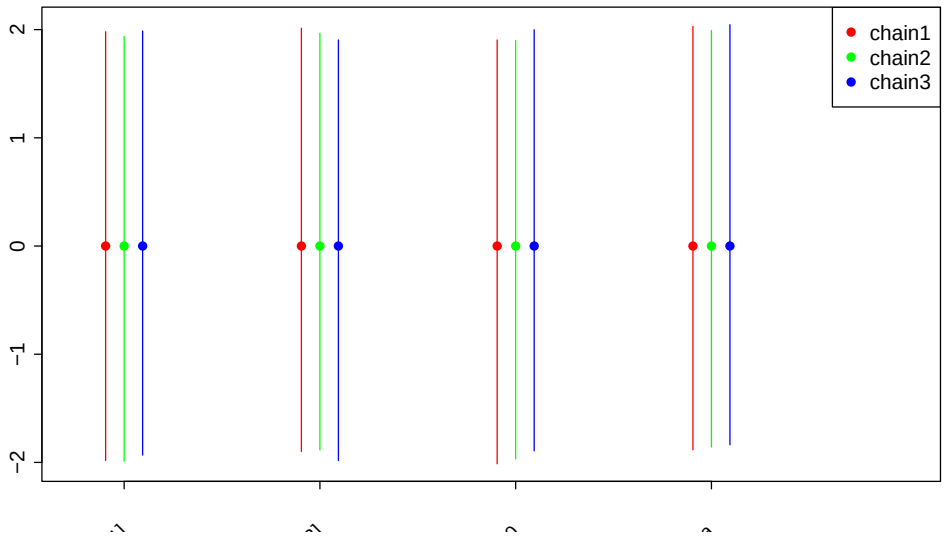

```
chainsSummary(samplesList) #dobimo mediane in credible intervals
```




```
chainsSummary(samplesList, scale = TRUE) #normaliziramo, tako da vse na isti skali
```




```
# V tem ukazu ni parametra burnin, zato to naredimo rocno:
samplesList2 <- samplesList
for (i in 1:3) samplesList2[[i]] <- samplesList2[[i]][2001:12000,]
chainsSummary(samplesList2, scale = TRUE)
```



2.3 Isti primer z izboljsano konvergenco

Bistvo:

- Centriranje in effective sample size - centriranje spremenljivk (odstejemo povprecje) nujno pri nimble.

- Centriranje spremenljivk (odstejemo povprecje pri vsaki neodvisni) ne spremeni regresijskih koeficientov, le regresijsko konstanto, zato interpretacija ostane enaka :)
- Standardizacija spremenljivk (odstejemo povprecje in delimo s standardnim odklonom pri vsaki neodvisni) pa bi spremenila vse regresijske koeficiente.
- Prednosti/slabosti bayesx in nimble.

Ob uporabi paketa BayesX:

```
fit2.bayesx <- bayesx(trening ~ kontrace + starost, data = sportnice, family = "gaussian
```

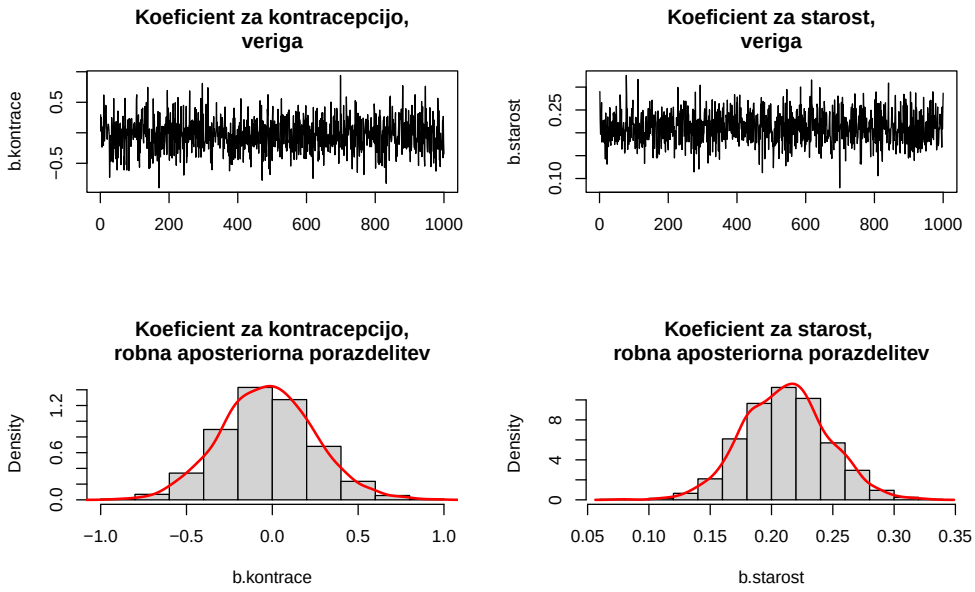
```
## Note: created new output directory 'C:/Users/ninar/AppData/Local/Temp/Rtmpclf3Ph/baye
```

```
summary(fit2.bayesx)
```

```
## Call:
## bayesx(formula = trening ~ kontrace + starost, data = sportnice,
##     family = "gaussian", method = "MCMC")
##
## Fixed effects estimation results:
##
## Parametric coefficients:
##                Mean      Sd    2.5%     50%  97.5%
## (Intercept)  3.7271  0.6126  2.4965  3.7319 4.9214
## kontrace    -0.0279  0.2705 -0.5549 -0.0305 0.5264
## starost      0.2113  0.0343  0.1448  0.2114 0.2801
##
## Scale estimate:
##          Mean     Sd   2.5%    50%  97.5%
## Sigma2 7.8063 0.4440 6.9998 7.7880 8.7487
##
## N = 660  burnin = 2000  method = MCMC  family = gaussian
## iterations = 12000  step = 10
```

```
b.kontrace <- attr(fit2.bayesx$fixed.effects, "sample")[,2]
b.starost <- attr(fit2.bayesx$fixed.effects, "sample")[,3]
```

```
par(mfrow = c(2, 2))
plot(b.kontrace, type = "l", main = "Koeficient za kontracepcijo,\nveriga",
     xlab = "")
plot(b.starost, type = "l", main = "Koeficient za starost,\nveriga",
     xlab = "")
hist(b.kontrace, prob = T, main = "Koeficient za kontracepcijo,\nrobna aposteriorna pora
lines(density(b.kontrace), col = "red", lwd = 2)
hist(b.starost, prob = T, main = "Koeficient za starost, \nrobna aposteriorna porazdelit
lines(density(b.starost), col = "red", lwd = 2)
```




```
par(mfrow = c(1, 1))
```

```
library(coda)
effectiveSize(b.kontrace) #1000
```

```
## var1
## 1000
```

```
effectiveSize(b.starost)  #1000
```

```
## var1
## 1000
```

```
?bayesx
```

```
## starting httpd help server ... done
```

```
?bayesx.control
# Torej je bil step=10 (thinning), burnin=2000, iterations=12000
# Dobili smo najvecji mozen effective sample size.

# Poskusimo, kaj se zgodi brez thinninga:
fit2.bayesx <- bayesx(trening ~ kontrace + starost, data = sportnice, family = "gaussian
                      step = 1)
```

```
## Note: created new output directory 'C:/Users/ninar/AppData/Local/Temp/Rtmpclf3Ph/baye
```

```
b.kontrace <- attr(fit2.bayesx$fixed.effects, "sample")[,2]
b.starost <- attr(fit2.bayesx$fixed.effects, "sample")[,3]

effectiveSize(b.kontrace) #10000, kar je najvecje mozno
```

```
##  var1
## 10000
```

```
effectiveSize(b.starost)  #10000, kar je najvecje mozno
```

```
##  var1
## 10000
```

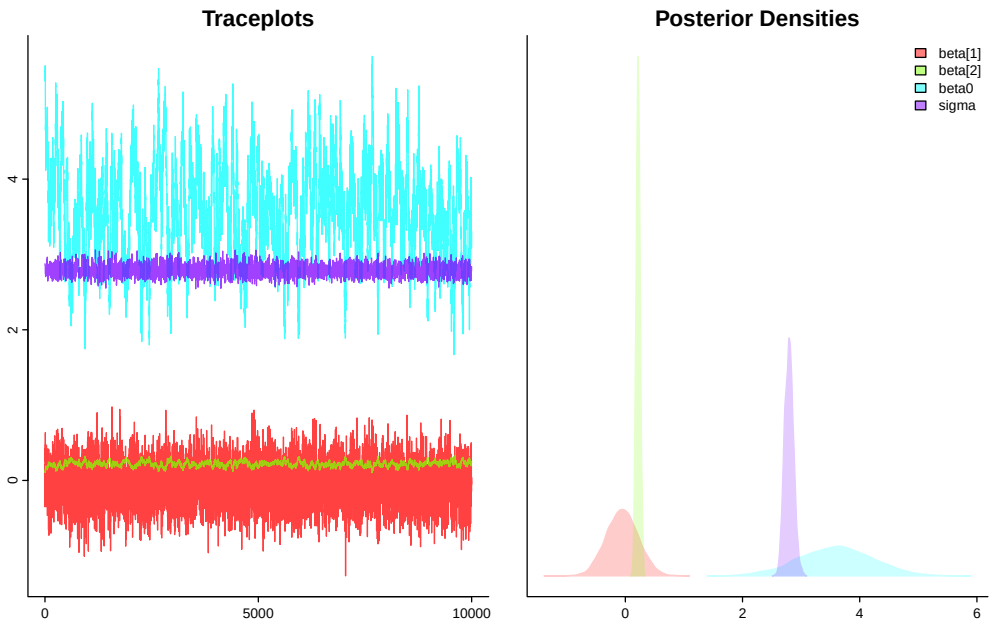
```
#Prednost bayesx: odlicno implementirani in izbrani samplerji, prilagojeno "obicajnim"
#Slabost bayesx: ne moremo spreminjati apriornih, specificirati kaksne drugacne modele
```

Bayesovski pristop s paketom nimble: - smo ze generirali verigo z 10000 iteracijami (shranjeno v sampels) - smo ze generirali tri verige s po 10000 iteracijami (shranjeno v samplesList):

```
samplesSummary(samples)
```

```
##                Mean      Median    St.Dev.  95%CI_low 95%CI_upp
## beta[1] -0.05189706 -0.05288237 0.27393993 -0.5764737 0.4839261
## beta[2]  0.21898917  0.21895438 0.03504209  0.1513766 0.2898851
## beta0    3.59040371  3.59879998 0.62281456  2.3377187 4.7996620
## sigma    2.79405128  2.79363184 0.07722947  2.6494137 2.9494305
```

```
samplesPlot(samples)
```




```
effectiveSize(samples) #obupno majhni
```

```
##   beta[1]   beta[2]     beta0     sigma
## 2083.3441  155.9336  162.6634 2220.8493
```

```
cor(samples) #razlog je koreliranost koeficientov, resitev je centriranje
```

```
##             beta[1]     beta[2]       beta0       sigma
```

```
## beta[1]  1.00000000 -0.32660145  0.23332349 -0.01216528
## beta[2] -0.32660145  1.00000000 -0.98037686  0.02462909
## beta0    0.23332349 -0.98037686  1.00000000 -0.02650145
## sigma   -0.01216528  0.02462909 -0.02650145  1.00000000
```

```
# Pozenemo isti model na centriranih podatkih:
X.centr = apply(X, 2, function(y) y - mean(y))

constants.centr <- list(n = length(sportnice$trening),
                        p = p,
                        x = X.centr)

Rmodel.centr <- nimbleModel(code, constants.centr, data, inits)
```

```
## Defining model

## Building model

## Setting data and initial values

## Running calculate on model
##   [Note] Any error reports that follow may simply reflect missing values in model var

## Checking model sizes and dimensions
```

```
conf.centr <- configureMCMC(Rmodel.centr)
```

```
## ===== Monitors =====
## thin = 1: beta, beta0, sigma
## ===== Samplers =====
## RW sampler (1)
##   - sigma
## conjugate sampler (3)
##   - beta0
##   - beta[]  (2 elements)
```

```
Rmcmc.centr <- buildMCMC(conf.centr)
Cmodel.centr <- compileNimble(Rmodel.centr)
```

```
## Compiling
##   [Note] This may take a minute.
##   [Note] Use 'showCompilerOutput = TRUE' to see C++ compilation details.
```

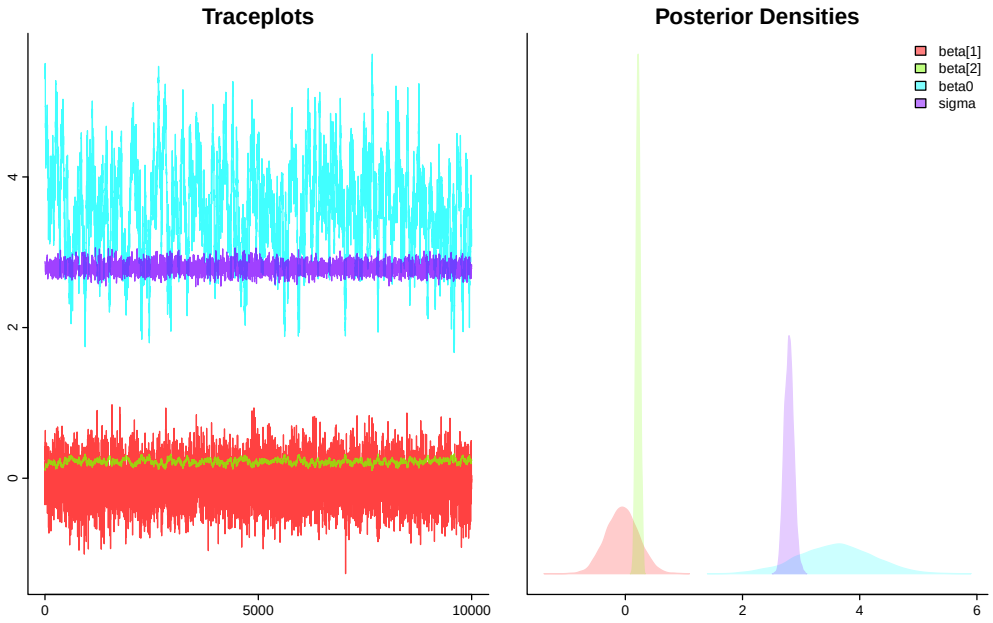
```
Cmcmc.centr <- compileNimble(Rmcmc.centr, project = Cmodel.centr)
```

```
## Compiling
##   [Note] This may take a minute.
##   [Note] Use 'showCompilerOutput = TRUE' to see C++ compilation details.
```

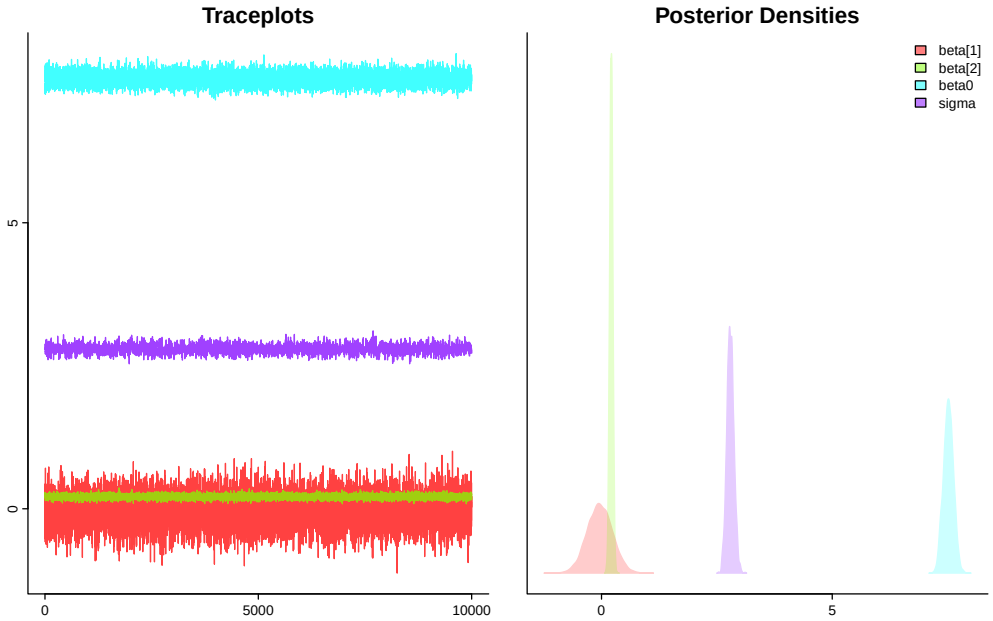
```
samples.centr <- runMCMC(Cmcmc.centr, niter = 12000, nburnin = 2000)
```

```
## running chain 1...
## |-------------|-------------|-------------|-------------|
## |-------------------------------------------------------|
```

```
samplesPlot(samples)        #pred centriranjem
```




```
samplesPlot(samples.centr) #po centriranju
```



```
effectiveSize(samples.centr) #zelo dobro
```

```
##   beta[1]   beta[2]     beta0     sigma
##  8470.059  8429.194 10000.000  2181.551
```

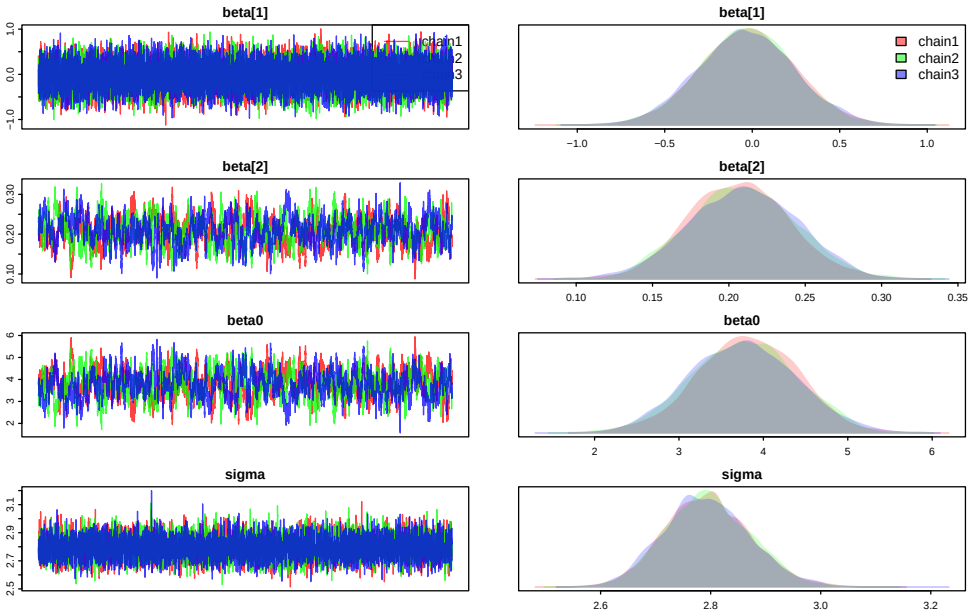
```
cor(samples.centr) #veliko manjse kot prej
```

```
##              beta[1]      beta[2]        beta0        sigma
## beta[1]  1.000000000 -0.312711683 -0.009473888 -0.001803840
## beta[2] -0.312711683  1.000000000 -0.008792551 -0.008024809
## beta0   -0.009473888 -0.008792551  1.000000000  0.011339690
## sigma   -0.001803840 -0.008024809  0.011339690  1.000000000
```

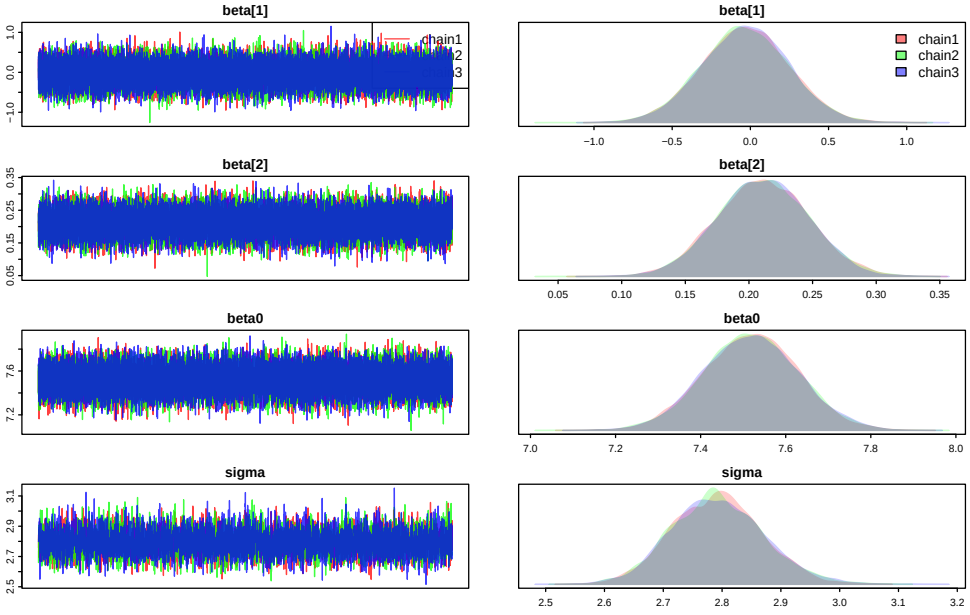
```
# Vec verig:
samplesList.centr <- runMCMC(Cmcmc.centr, niter = 12000, nburnin = 0,
                             nchains = 3, inits = initsFunction)
```

```
## running chain 1...
## |-------------|-------------|-------------|-------------|
## |-------------------------------------------------------|
## running chain 2...
## |-------------|-------------|-------------|-------------|
## |-------------------------------------------------------|
## running chain 3...
## |-------------|-------------|-------------|-------------|
## |-------------------------------------------------------|
```

```
chainsPlot(samplesList, burnin = 2000) #prej
```




```
chainsPlot(samplesList.centr, burnin = 2000) #sedaj
```




```
#Prednost nimble: popolna svoboda (model, apriorne, samplerji)
#Slabost nimble: potrebujemo nekaj znanja, da dobro izberemo
```

2.4 Model za povezanost kolicine treninga s tezo, visino in bmi

Z modelom multiple linearne regresije preverite, ali je kolicina treninga povezana s tezo, visino in bmi. Uporabite nimble. Generirajte vec verig in preucite konvergenco. Komentirajte rezultate.

```
code <- nimbleCode({
  beta0 ~ dnorm(0, sd = 100)
  for(k in 1:p) { #lahko uporabimo for zanko za definicijo vseh apriornih porazdelitev
    beta[k] ~ dnorm(0, sd = 100)
  }
  sigma ~ dunif(0, 100)
  for(i in 1:n) {
    y[i] ~ dnorm(beta0 + inprod(beta[1:p], x[i, 1:p]), sd = sigma) #uporabimo funkcijo
  }
})

X <- subset(sportnice, select = c("teza",
                                  "visina",
                                  "bmi"))
X = apply(X, 2, function(y) y - mean(y))
p <- ncol(X)

constants <- list(n = length(sportnice$trening),
                  p = p,
                  x = X)

data <- list(y = sportnice$trening)

inits <- list(beta0 = mean(sportnice$trening),
              beta = rep(0, p),
              sigma = 1)
```

```
Rmodel <- nimbleModel(code, constants, data, inits)
```

```
## Defining model

## Building model

## Setting data and initial values

## Running calculate on model
##   [Note] Any error reports that follow may simply reflect missing values in model var

## Checking model sizes and dimensions
```

```
conf <- configureMCMC(Rmodel)
```

```
## ===== Monitors =====
## thin = 1: beta, beta0, sigma
## ===== Samplers =====
## RW sampler (1)
##   - sigma
```

```
## conjugate sampler (4)
##   - beta0
##   - beta[]  (3 elements)
```

```
Rmcmc <- buildMCMC(conf)
Cmodel <- compileNimble(Rmodel)
```

```
## Compiling
##   [Note] This may take a minute.
##   [Note] Use 'showCompilerOutput = TRUE' to see C++ compilation details.
```

```
Cmcmc <- compileNimble(Rmcmc, project = Cmodel)
```

```
## Compiling
##   [Note] This may take a minute.
##   [Note] Use 'showCompilerOutput = TRUE' to see C++ compilation details.
```

```
samples <- runMCMC(Cmcmc, niter = 12000, nburnin = 2000)
```

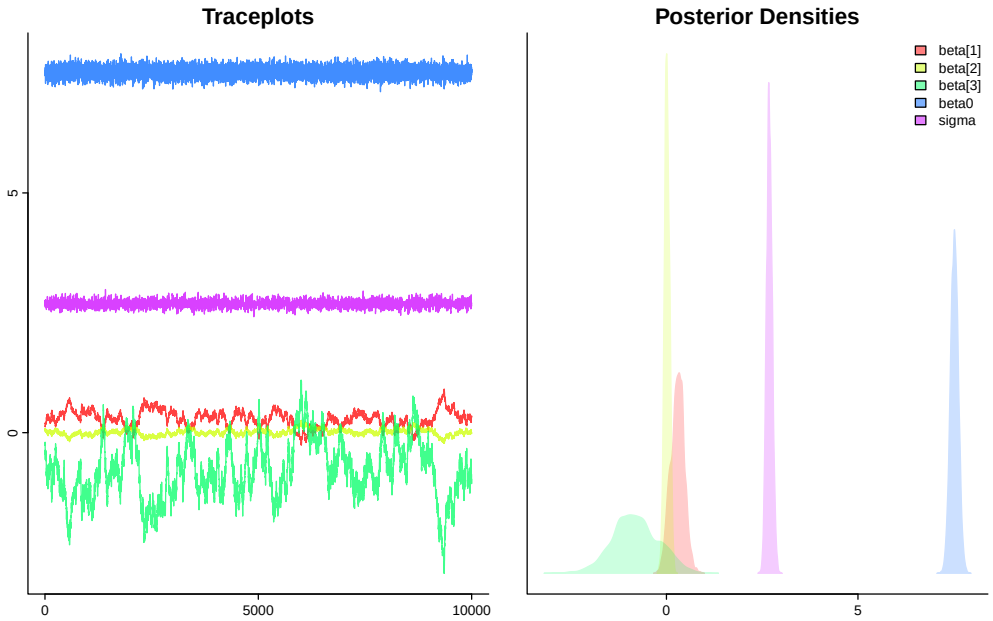
```
## running chain 1...
```

```
## |-------------|-------------|-------------|-------------|
## |-------------------------------------------------------|
```

```
samplesSummary(samples)
```

```
##                Mean     Median    St.Dev.   95%CI_low 95%CI_upp
## beta[1]  0.28833454  0.2964341 0.17616141 -0.05635164 0.6274081
## beta[2]  0.01570644  0.0128745 0.06844556 -0.11691358 0.1510380
## beta[3] -0.80104758 -0.8271471 0.60573140 -1.98110752 0.3859088
## beta0    7.51885993  7.5189657 0.10338899  7.31558104 7.7219434
## sigma    2.68843215  2.6854519 0.07538185  2.54569839 2.8400923
```

```
samplesPlot(samples)
```




```
effectiveSize(samples)
```

```
##     beta[1]     beta[2]     beta[3]       beta0       sigma
##    29.69440    29.75821    28.60660 10000.00000  2294.20836
```

```
cor(samples)
```

```
##              beta[1]      beta[2]     beta[3]        beta0        sigma
## beta[1]  1.000000000 -0.968445111 -0.99566498  0.009810336  0.025271571
## beta[2] -0.968445111  1.000000000  0.95607379 -0.004994208 -0.022545190
## beta[3] -0.995664978  0.956073794  1.00000000 -0.010014737 -0.024681549
## beta0    0.009810336 -0.004994208 -0.01001474  1.000000000  0.005578794
## sigma    0.025271571 -0.022545190 -0.02468155  0.005578794  1.000000000
```

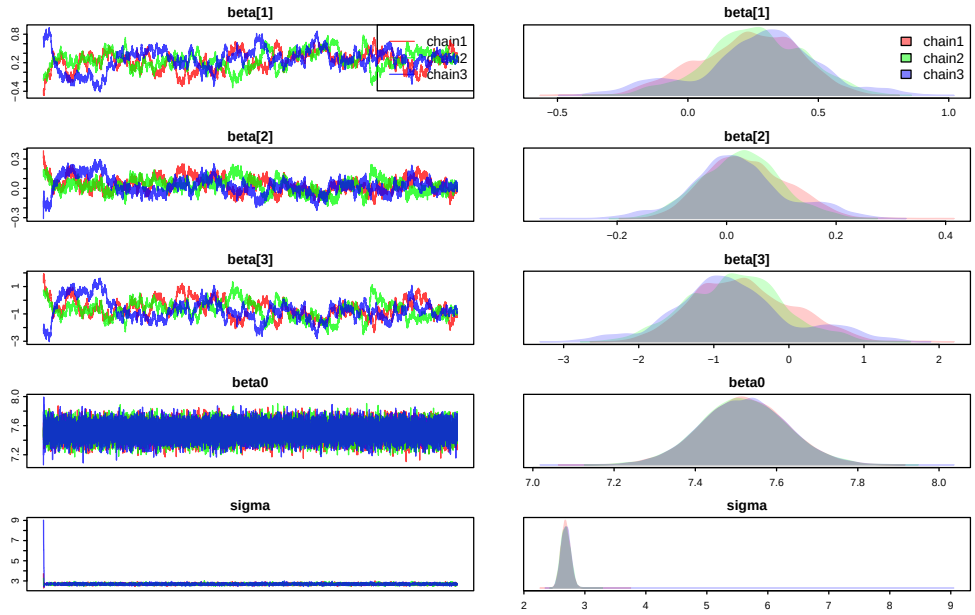
Vec verig z nakljucnimi zacetnimi vrednostmi:

```
initsFunction <- function(){
  list(beta0 = rnorm(1),
       beta = rnorm(3),
       sigma = runif(1, min = 0, max = 10))
}

samplesList <- runMCMC(Cmcmc, niter = 12000, nburnin = 0,
                       nchains = 3, inits = initsFunction)
```

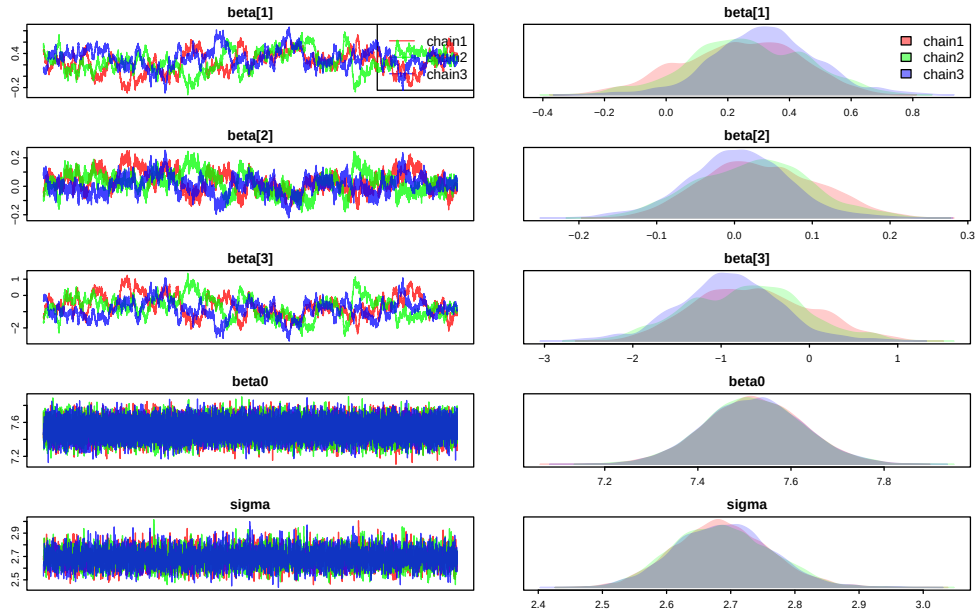
```
## running chain 1...
## |-------------|-------------|-------------|-------------|
## |-------------------------------------------------------|
## running chain 2...
```

```
## |-------------|-------------|-------------|-------------|
## |-------------------------------------------------------|

## running chain 3...

## |-------------|-------------|-------------|-------------|
## |-------------------------------------------------------|
```

```
chainsPlot(samplesList)
```




```
chainsPlot(samplesList, burnin = 2000)
```



Naloga: Komentirajte rezultate.

Opazimo: - effective sampel size so zelo majhni - korelacije med parametri so zelo velike - verige ne izgledajo v redu (avtokoreliranost)

Bistvo: Zaradi multikolinearnosti model ni dobro definiran, vidimo v nenavadnem obnasanju verig in effective sample size. Tu imamo ekstremen primer multikolinearnosti, saj je bmi s formulo izracunan iz teze in visine - sprasujemo se nemogoce vprasanje: Kaksna je razlika v kolicini treninga za dve sportnici, ki sta enako visoki in tezki ter imata razlicen bmi? Ker taksne sportnice v nasih podatkih ne obstajajo, tega ne moremo oceniti.

8. sklop: Hierarhicni (regresijski) modeli

Tu predstavimo celotno generirano poročilo iz izvorne datoteke, vendar študentom ne priporočamo generiranje celotnega poročila iz izvorne datoteke, saj bo to trajalo dolgo časa (veliko modelov, veliko iteracij) - raje preglejte in prevedite izvorno kodo po delih.

```
# Nalozimo pakete za nimble in preucevanje konvergence:
library(nimble)
```

```
## nimble version 1.0.1 is loaded.
## For more information on NIMBLE and a User Manual,
## please visit https://R-nimble.org.
##
## Note for advanced users who have written their own MCMC samplers:
##   As of version 0.13.0, NIMBLE's protocol for handling posterior
##   predictive nodes has changed in a way that could affect user-defined
##   samplers in some situations. Please see Section 15.5.1 of the User Manual.
##
## Attaching package: 'nimble'
## The following object is masked from 'package:stats':
##
##     simulate
```

```
library(basicMCMCplots)
library(coda)
```

1 Hierarhicni normalni model z enakimi variancami - sole se enkrat

Na primer s solami lahko gledamo tudi kakor na 100 regresijskih modelov (za vsako solo eden): rezultatMatTesta ~ 1 (tj. ocenjujemo le regresijsko konstanto), ki so **skupaj povezani s hiperparametri**.

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(reshape2)
source("podatki_sole.R")
str(pod)
```

```
## 'data.frame':    1993 obs. of  2 variables:
##  $ school   : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ mathscore: num  52.1 57.6 66.4 44.7 40.6 ...
```

```
pod.sole = pod %>%
  group_by(school) %>%
  summarise(povprecje = mean(mathscore), n=length(mathscore), varianca = var(mathscore))
str(pod.sole)
```

```
## tibble [100 x 4] (S3: tbl_df/tbl/data.frame)
##  $ school   : int [1:100] 1 2 3 4 5 6 7 8 9 10 ...
##  $ povprecje: num [1:100] 50.8 46.5 48.8 47.3 36.6 ...
##  $ n        : int [1:100] 31 22 23 19 21 16 16 22 24 18 ...
##  $ varianca : num [1:100] 126.6 98.3 58.2 112.8 69.5 ...
```

```
# Rezultate vsake sole damo v svoj stolpec.
# Ker imamo po solah razlicno velike vzorce, bodo imeli nekateri stolpci na koncu NA (
m <- length(pod.sole$school)
n <- pod.sole$n
yMatrix <- matrix(NA, ncol = m, nrow = max(n))
for (j in 1:m) {
  yMatrix[1:n[j],j] <- pod[pod$school==j,]$mathscore
}

View(yMatrix)
```

```
code <- nimbleCode({
  mu ~ dnorm(0, sd = 100); #apriorna za hiperparameter
  eta ~ dunif(0, 100)      #apriorna za hiperparameter
  sigma ~ dunif(0, 100)    #apriorna za parameter
  #za vse (hiper)apriorne smo izbrali nekaj zelo neinformativnega
  #opazimo: nismo se obremenjevali s primernostjo druzine porazdelitve (npr. inverzna

  for (j in 1:m) {
    muGroups[j] ~ dnorm(mu, sd = eta) #porazdelitve parametrov
```

```
    for (i in 1:n[j]) {
      y[i, j] ~ dnorm(muGroups[j], sd = sigma); #model
    }
  }
})

constants <- list(m = m, n = n)

inits <- list(mu = mean(pod.sole$povprecje), #tako kot pri sklopu z Gibbsovim vzorceval
              eta = sd(pod.sole$povprecje),
              sigma = mean(sqrt(pod.sole$varianca)),
              muGroups = pod.sole$povprecje)

data <- list(y = yMatrix)
```

```
Rmodel <- nimbleModel(code = code, constants = constants,
                      inits = inits, data = data) #smo dobili opozorilo, ne error (tudi
```

```
## Defining model

## Building model

## Setting data and initial values

## Running calculate on model
##   [Note] Any error reports that follow may simply reflect missing values in model var

## Checking model sizes and dimensions

##   [Note] This model is not fully initialized. This is not an error.
##          To see which variables are not initialized, use model$initializeInfo().
##          For more information on model initialization, see help(modelInitialization).
```

```
Rmodel$initializeInfo() #izvemo, da je bilo opozorilo zaradi NA v podatkih - bo vseeno
```

```
##   [Note] Missing values (NAs) or non-finite values were found in model variables: y.
##   [Note] This is not an error, but some or all variables may need to be initialized f
##   [Note] For more information on model initialization, see help(modelInitialization).
```

```
conf <- configureMCMC(Rmodel) #vzorcenj za parametre mu_i si ne bo avtomatsko zapomnil
```

```
## ===== Monitors =====
## thin = 1: eta, mu, sigma
## ===== Samplers =====
## RW sampler (2)
##   - eta
##   - sigma
## conjugate sampler (101)
##   - mu
```

```
##   - muGroups[]  (100 elements)
```

```
conf$addMonitors('muGroups') #dodamo shranjevanje muGroups
```

```
## thin = 1: eta, mu, muGroups, sigma
```

```
conf$printMonitors() #je dodano
```

```
## thin = 1: eta, mu, muGroups, sigma
```

```
Rmcmc <- buildMCMC(conf)
Cmodel <- compileNimble(Rmodel)
```

```
## Compiling
##   [Note] This may take a minute.
##   [Note] Use 'showCompilerOutput = TRUE' to see C++ compilation details.
```

```
Cmcmc <- compileNimble(Rmcmc, project = Cmodel)
```

```
## Compiling
##   [Note] This may take a minute.
##   [Note] Use 'showCompilerOutput = TRUE' to see C++ compilation details.
```

```
samples <- runMCMC(Cmcmc, niter = 12000, nburnin = 2000)
```

```
## running chain 1...
```

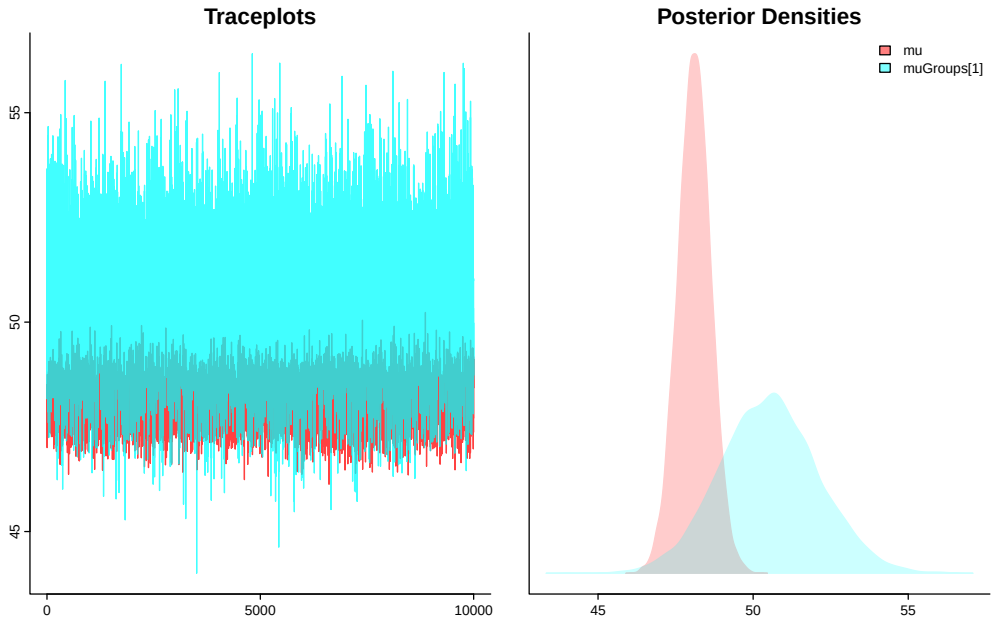
```
## |-------------|-------------|-------------|-------------|
## |-------------------------------------------------------|
```

O rezultatih:

```
# Si ogledamo nekatere (hiper)parametre
samplesSummary(samples)[c(2, 3), ]
```

```
##                  Mean   Median   St.Dev. 95%CI_low 95%CI_upp
## mu           48.10386 48.10617 0.5352283  47.04270  49.15175
## muGroups[1] 50.51351 50.50845 1.5832888  47.43126  53.67767
```

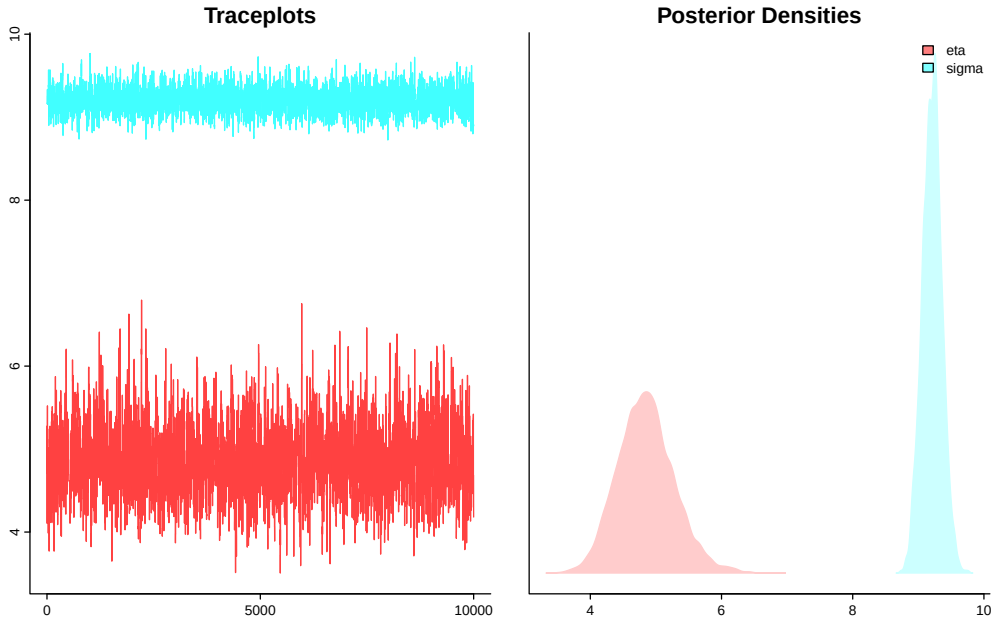
```
samplesPlot(samples, var = c("mu","muGroups[1]"))
```




```
samplesSummary(samples)[c(1, 103), ]
```

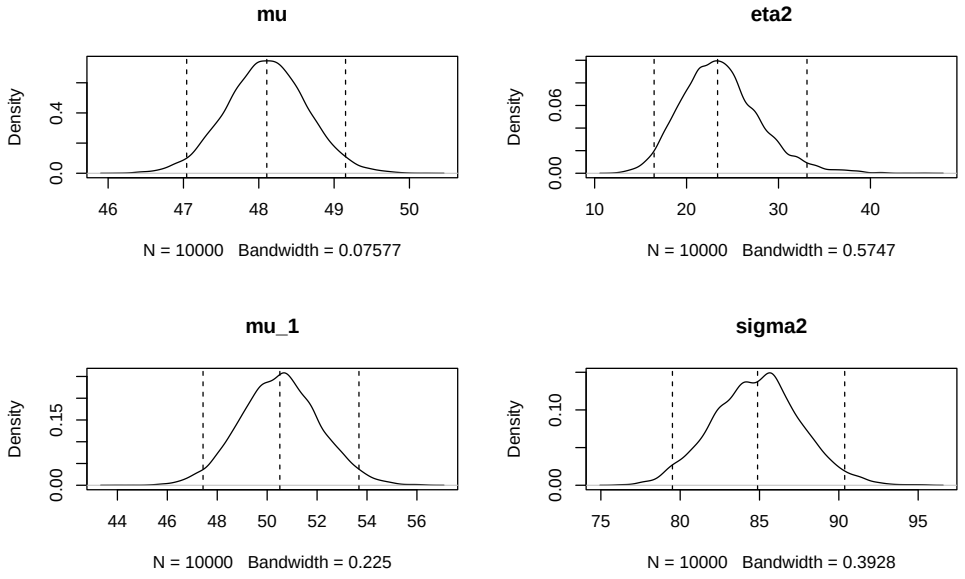
```
##           Mean   Median   St.Dev. 95%CI_low 95%CI_upp
## eta   4.849233 4.834427 0.4307974  4.058082  5.753105
## sigma 9.210310 9.212957 0.1494535  8.916911  9.506340
```

```
samplesPlot(samples, var = c("eta","sigma"))
```




```
# Preko spodnje slike opazimo, da so nasi rezultati enaki/podobni kakor v 6. sklopu z
par(mfrow=c(2, 2))
plot(density(samples[ , 2]), type = "l", main = "mu")
abline(v = quantile(samples[ , 2], prob=c(0.025, 0.5, 0.975)), lty = 2)
```

```
plot(density(samples[ , 1]**2), type = "l", main = "eta2")
abline(v = quantile(samples[ , 1]**2, prob=c(0.025, 0.5, 0.975)), lty = 2)
plot(density(samples[ , 3]), type = "l", main = "mu_1")
abline(v = quantile(samples[ , 3], prob=c(0.025, 0.5, 0.975)), lty = 2)
plot(density(samples[ , 103]**2), type = "l", main = "sigma2")
abline(v = quantile(samples[ , 103]**2, prob=c(0.025, 0.5, 0.975)), lty = 2)
```




```
par(mfrow = c(1, 1))
```

O konvergenci:

```
effectiveSize(samples)
```

```
##           eta            mu   muGroups[1]   muGroups[2]   muGroups[3]
##      1321.028      7026.935     10000.000     10000.000     10000.000
##   muGroups[4]   muGroups[5]   muGroups[6]   muGroups[7]   muGroups[8]
##     10113.824      9199.054      9698.404      9827.518     10000.000
##   muGroups[9]  muGroups[10]  muGroups[11]  muGroups[12]  muGroups[13]
##     10000.000     10000.000      9713.087     10000.000     10000.000
##  muGroups[14]  muGroups[15]  muGroups[16]  muGroups[17]  muGroups[18]
##      9166.351      8896.424      9198.409      9241.893     10000.000
##  muGroups[19]  muGroups[20]  muGroups[21]  muGroups[22]  muGroups[23]
##     10133.166     10000.000     10000.000     10000.000     10000.000
##  muGroups[24]  muGroups[25]  muGroups[26]  muGroups[27]  muGroups[28]
##     10000.000     10000.000     10000.000     10000.000     10000.000
##  muGroups[29]  muGroups[30]  muGroups[31]  muGroups[32]  muGroups[33]
##     10000.000     10000.000      9577.018      9541.898      9351.686
##  muGroups[34]  muGroups[35]  muGroups[36]  muGroups[37]  muGroups[38]
##      9348.340      9847.620      9605.346     10000.000     10520.189
```

```
##  muGroups[39]  muGroups[40]  muGroups[41]  muGroups[42]  muGroups[43]
##     10000.000     10000.000      9315.889      9509.162      9550.327
##  muGroups[44]  muGroups[45]  muGroups[46]  muGroups[47]  muGroups[48]
##     10000.000     10000.000      9554.280     10000.000     10000.000
##  muGroups[49]  muGroups[50]  muGroups[51]  muGroups[52]  muGroups[53]
##      9443.935     10000.000      8331.256     10000.000      9193.366
##  muGroups[54]  muGroups[55]  muGroups[56]  muGroups[57]  muGroups[58]
##     10000.000     10000.000     10000.000     10000.000      9701.306
##  muGroups[59]  muGroups[60]  muGroups[61]  muGroups[62]  muGroups[63]
##     10000.000     10000.000     10000.000      9521.974      9693.085
##  muGroups[64]  muGroups[65]  muGroups[66]  muGroups[67]  muGroups[68]
##      9971.069      9169.995     10000.000      7182.014     10278.401
##  muGroups[69]  muGroups[70]  muGroups[71]  muGroups[72]  muGroups[73]
##     10000.000     10000.000     10000.000      8591.005     10000.000
##  muGroups[74]  muGroups[75]  muGroups[76]  muGroups[77]  muGroups[78]
##      9645.944     10000.000      9705.953     10000.000      9594.335
##  muGroups[79]  muGroups[80]  muGroups[81]  muGroups[82]  muGroups[83]
##      7913.348     10000.000     10000.000      9579.069     10000.000
##  muGroups[84]  muGroups[85]  muGroups[86]  muGroups[87]  muGroups[88]
##     10000.000     10000.000     10000.000     10000.000     10000.000
##  muGroups[89]  muGroups[90]  muGroups[91]  muGroups[92]  muGroups[93]
##      9674.816     10000.000     10000.000     10000.000     10000.000
##  muGroups[94]  muGroups[95]  muGroups[96]  muGroups[97]  muGroups[98]
##     10000.000     10000.000     10000.000      9603.400     10000.000
##  muGroups[99] muGroups[100]         sigma
##     10289.006     10000.000      2197.510
```

```
head(
  sort(
    effectiveSize(samples))) #najmanjsi izmed effective sample size - nekako ok (slabs
```

```
##          eta        sigma           mu muGroups[67] muGroups[79] muGroups[51]
##     1321.028     2197.510     7026.935     7182.014     7913.348     8331.256
```

```
#cor(samples)  neuporabno, ker je izpis predolg
```

```
head(
  sort(
    abs(
      cor(samples)[cor(samples)!=1]), decreasing = T)) #ok
```

```
## [1] 0.2167157 0.2167157 0.1946722 0.1946722 0.1923242 0.1923242
```

```
# Pogledamo se vec verig:
initsFunction <- function(){ #nakljucno generiranje zacetnih vrednosti (podobno kot v p
  list(mu = rnorm(1, mean = mean(pod.sole$povprecje), sd = 10),
       eta = runif(1, min = 0, max = 10),
       sigma = runif(1, min = 0, max = 10),
       muGroups = rnorm(m, mean = pod.sole$povprecje, sd = 10))
}

samplesList <- runMCMC(Cmcmc, niter = 12000, nburnin = 2000,
                       nchains = 3, inits = initsFunction)
```

```
## running chain 1...
```

```
## |-------------|-------------|-------------|-------------|
## |-------------------------------------------------------|
```

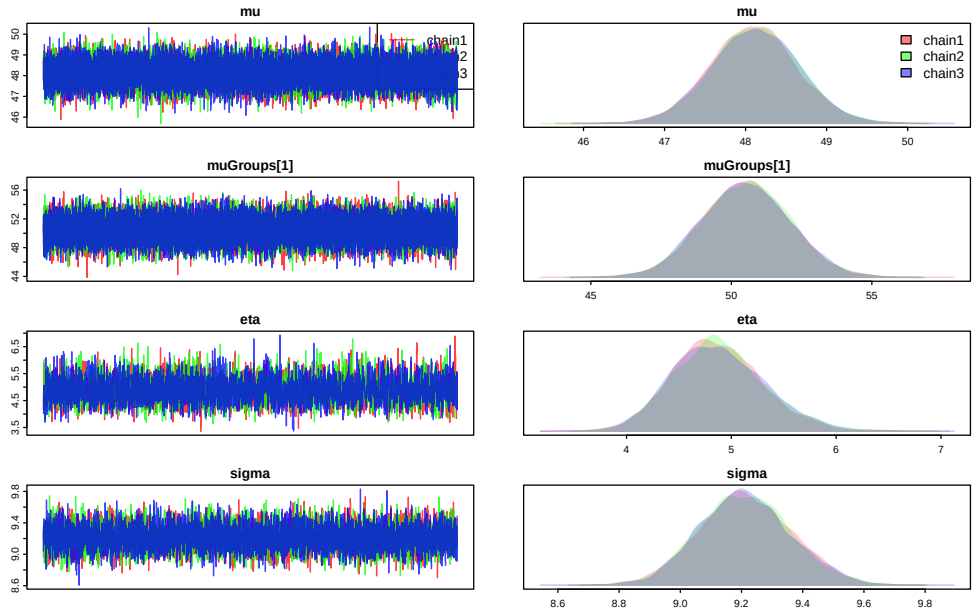
```
## running chain 2...
```

```
## |-------------|-------------|-------------|-------------|
## |-------------------------------------------------------|
```

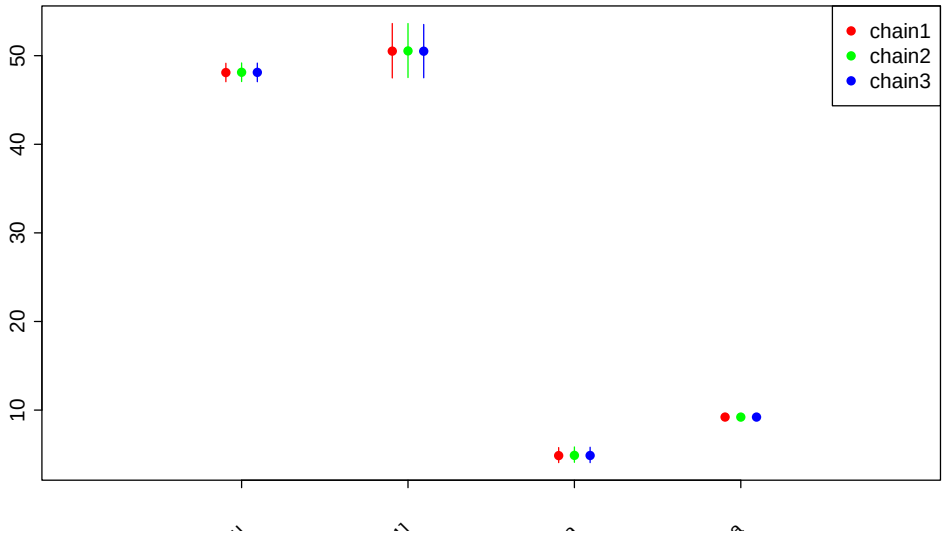
```
## running chain 3...
```

```
## |-------------|-------------|-------------|-------------|
## |-------------------------------------------------------|
```

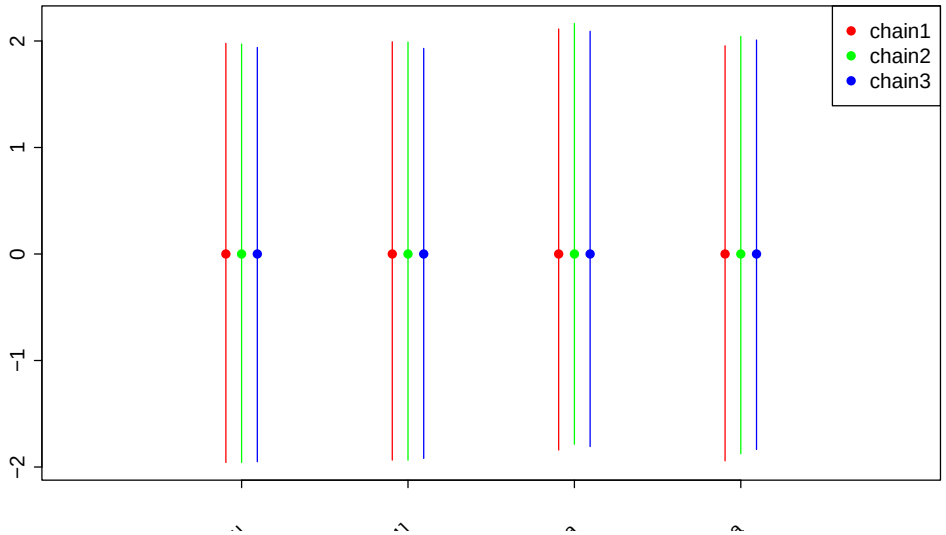
```
chainsPlot(samplesList, var = c("mu", "muGroups[1]", "eta", "sigma")) #dobro
```




```
chainsSummary(samplesList, buffer.left = 1, buffer.right = 1, scale = FALSE,
              var = c("mu", "muGroups[1]", "eta", "sigma"))
```




```
chainsSummary(samplesList, buffer.left = 1, buffer.right = 1, scale = TRUE, #dobro
              var = c("mu", "muGroups[1]", "eta", "sigma"))
```



Bistvo: Dobili smo zelo podobne rezultate kakor z Gibbsovim vzorcevalnikom, vendar nismo potrebovali teoreticnih izpeljav. Konvergenca je bila dobra, pri cemer se nam ni bilo treba ukvarjati z izborom samplerjev. Ni pa nujno, da je vedno tako (kaksna koreliranost parametrov nas lahko prisili v drugacen izbor samplerjev, itd.).

2 Hierarhicni regresijski model (z enakimi variancami) - podatki o solah z dodatno spremenljivko

V novih podatkih imamo dodano spremenljivko SES, tj. socio-ekonomski status. Ta je bila izracunana iz dohodka starsev in njihove izobrazbe.

Zanima nas povezanost rezultata matematicnega testa s SES.

Naredimo po eno linearno regresijo: rezultatMatTesta ~ SES za vsako solo -> dobimo mu_j (regresijska konstanta) in beta_j za vsako solo j, kjer vse mu_j povezemo hierarhicno (kot prej), enako pa naredimo tudi za beta_j.

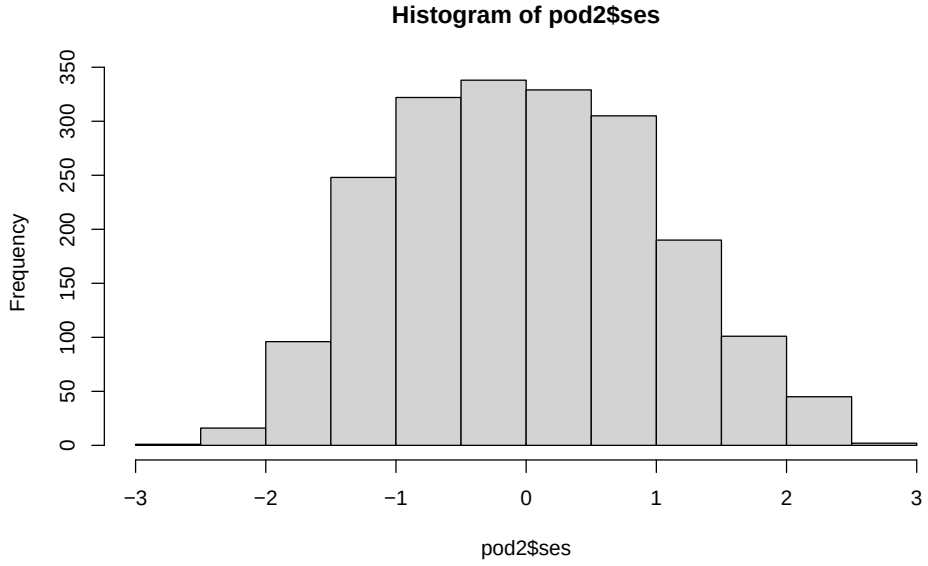
```
source("podatki_sole2.R")
str(pod2)
```

```
## 'data.frame':    1993 obs. of  3 variables:
##  $ school   : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ mathscore: num  52.1 57.6 66.4 44.7 40.6 ...
##  $ ses      : num  -0.0599 1.0517 -0.8635 -0.7966 -1.6135 ...
```

```
max(abs(pod - pod2[,c(1,2)])) #isto kot prej, le SES je dodan - lahko ohranimo ze narej
```

```
## [1] 0
```

```
hist(pod2$ses)
```




```
# Tako kot smo prej shranili rezultat matematicnega testa v yMatrix, naredimo sedaj se
xMatrix <- matrix(NA, ncol = m, nrow = max(n))
for (j in 1:m) {
  xMatrix[1:n[j],j] <- pod2[pod2$school==j,]$ses - mean(pod2[pod2$school==j,]$ses) ### c
}
```

```
View(xMatrix)
```

Naloga: Implementirajte model s pomocjo nimble tako, da predrugacite kodo prejsnjega modela brez SES. Prikazite rezultate in jih primerjajte s prejsnjim modelom brez SES. Preucite konvergenco (na podoben nacin kot pri prejsnjem modelu brez SES).

```
code2 <- nimbleCode({
  mu ~ dnorm(0, sd = 100) #hiperparameter za mu_j (kot prej)
  eta ~ dunif(0, 100)     #hiperparameter za mu_j (kot prej)
  sigma ~ dunif(0, 100)   #parameter za varianco ostankov pri linearni regresiji bo en

  beta ~ dnorm(0, sd = 100) #novo - hiperparameter za beta_j
  etaBeta ~ dunif(0, 100)   #novo - hiperparameter za beta_j

  for (j in 1:m) {
    muGroups[j] ~ dnorm(mu, sd = eta)
    betaGroups[j] ~ dnorm(beta, sd = etaBeta) #novo - povezava beta_j s hiperparametri
    for (i in 1:n[j]) {
      y[i, j] ~ dnorm(muGroups[j] + betaGroups[j] * x[i, j], sd = sigma); #v model smo
    }
  }
})

constants2 <- list(m = m, n = n)

inits2 <- list(mu = mean(pod.sole$povprecje),
               eta = sd(pod.sole$povprecje),
               sigma = mean(sqrt(pod.sole$varianca)),
               muGroups = pod.sole$povprecje,
               betaGroups = rep(0, m), #novo - nekaj vzamemo, ni pomembno (oz. ne sme b
               beta = 0,               #novo
               etaBeta = 1)            #novo

data2 <- list(y = yMatrix, x = xMatrix) #dodamo se xMatrix - lahko v data in ne v const
```

```
Rmodel2 <- nimbleModel(code = code2, constants = constants2,
                       inits = inits2, data = data2) #podobno opozorilo kot prej (zarad
```

```
## Defining model

## Building model

## Setting data and initial values

## Running calculate on model
##   [Note] Any error reports that follow may simply reflect missing values in model var
```

```
## Checking model sizes and dimensions
```

```
##   [Note] This model is not fully initialized. This is not an error.
##          To see which variables are not initialized, use model$initializeInfo().
##          For more information on model initialization, see help(modelInitialization).
```

```
Rmodel2$initializeInfo()
```

```
##   [Note] Missing values (NAs) or non-finite values were found in model variables: lif
##   [Note] This is not an error, but some or all variables may need to be initialized f
##   [Note] For more information on model initialization, see help(modelInitialization).
```

```
conf2 <- configureMCMC(Rmodel2)
```

```
## ===== Monitors =====
## thin = 1: beta, eta, etaBeta, mu, sigma
## ===== Samplers =====
## RW sampler (3)
##   - eta
##   - sigma
##   - etaBeta
## conjugate sampler (202)
##   - mu
##   - beta
##   - muGroups[]  (100 elements)
##   - betaGroups[]  (100 elements)
```

```
conf2$addMonitors('muGroups', 'betaGroups')   #dodamo shranjevanje mu_j in beta_j
```

```
## thin = 1: beta, betaGroups, eta, etaBeta, mu, muGroups, sigma
```

```
Rmcmc2 <- buildMCMC(conf2)
Cmodel2 <- compileNimble(Rmodel2)
```

```
## Compiling
##   [Note] This may take a minute.
##   [Note] Use 'showCompilerOutput = TRUE' to see C++ compilation details.
```

```
Cmcmc2 <- compileNimble(Rmcmc2, project = Cmodel2)
```

```
## Compiling
##   [Note] This may take a minute.
##   [Note] Use 'showCompilerOutput = TRUE' to see C++ compilation details.
```

```
samples2 <- runMCMC(Cmcmc2, niter = 12000, nburnin = 2000)
```

```
## running chain 1...
```

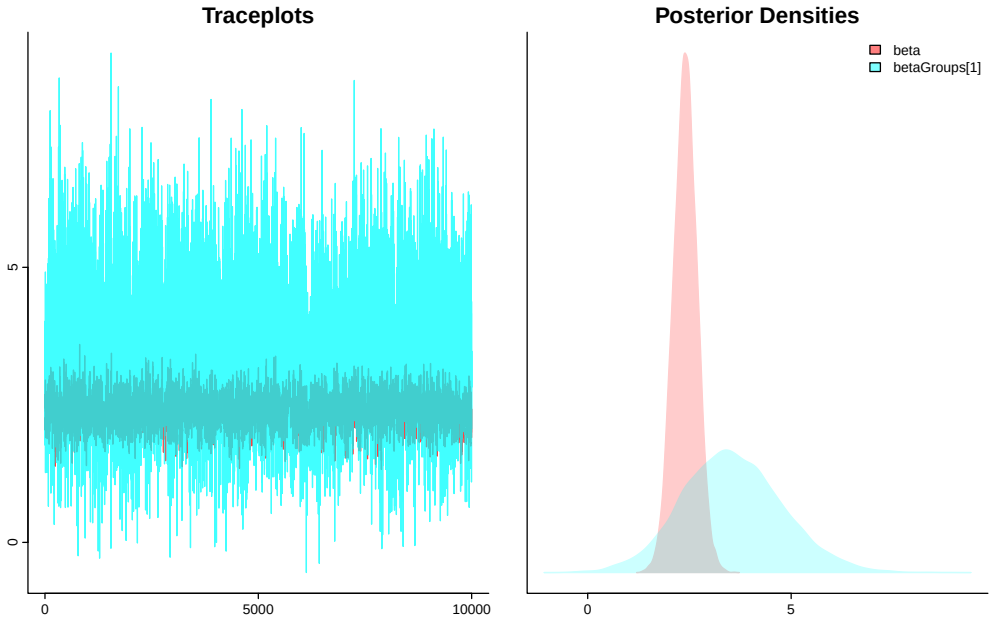
```
## |-------------|-------------|-------------|-------------|
## |-------------------------------------------------------|
```

O rezultatih:

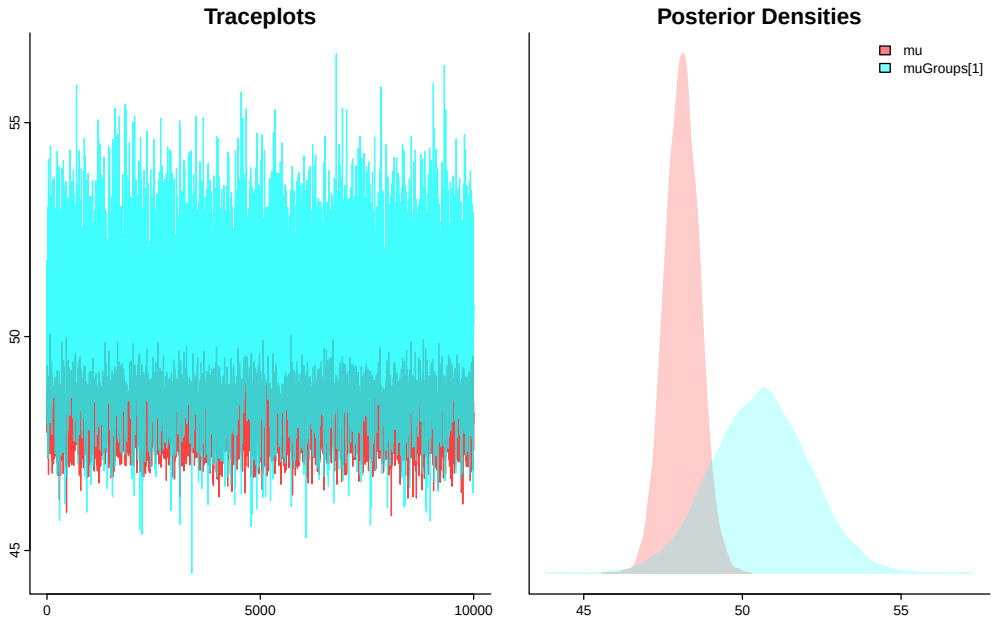
```
samplesSummary(samples2)[c(1, 103, 2, 104, 102, 105, 205),]
```

```
##                       Mean     Median   St.Dev.  95%CI_low 95%CI_upp
## beta              2.395171  2.397413 0.2896837  1.8231182  2.959655
## etaBeta           1.664909  1.660263 0.3618534  0.9704382  2.370622
## betaGroups[1]     3.555992  3.514532 1.2242308  1.2317228  6.070365
## mu               48.103856 48.102074 0.5366006 47.0552372 49.165067
## eta               4.925985  4.917058 0.4339132  4.1160437  5.829360
## muGroups[1]      50.566680 50.584665 1.5221464 47.5814686 53.574614
## sigma             8.817535  8.816525 0.1461253  8.5392818  9.106915
```

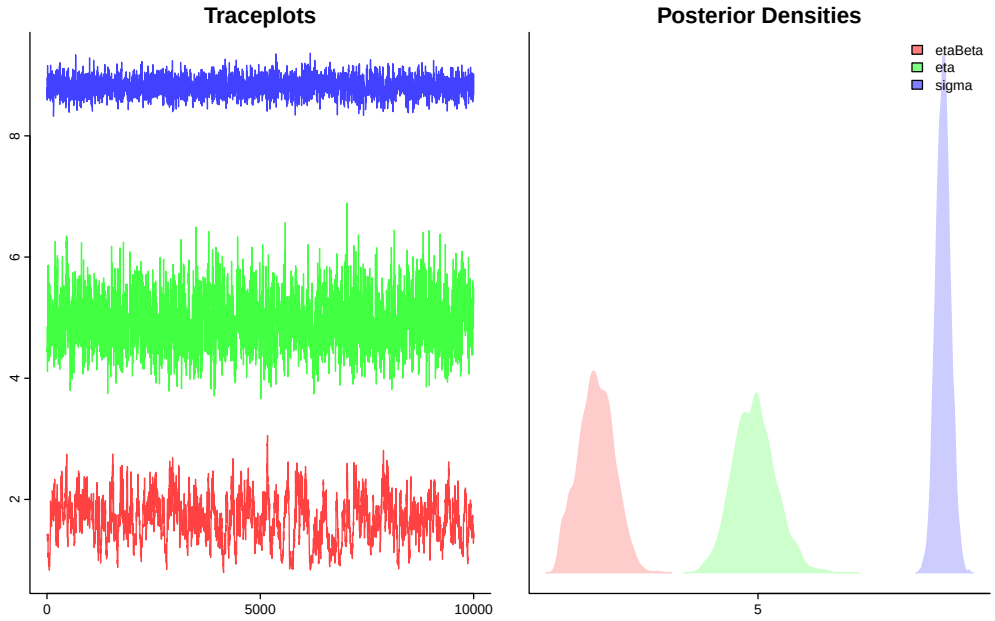
```
samplesPlot(samples2, var = c("beta","betaGroups[1]"))
```




```
samplesPlot(samples2, var = c("mu","muGroups[1]"))
```




```
samplesPlot(samples2, var = c("etaBeta","eta","sigma"))
```




```
samplesSummary(samples)[c(2, 1, 3, 103), ] #rezultati modela brez SES - so podobni
```

```
##                    Mean    Median   St.Dev. 95%CI_low 95%CI_upp
## mu            48.103856 48.106170 0.5352283 47.042695 49.151748
## eta            4.849233  4.834427 0.4307974  4.058082  5.753105
## muGroups[1] 50.513505 50.508448 1.5832888 47.431264 53.677672
## sigma          9.210310  9.212957 0.1494535  8.916911  9.506340
```

```
# Kaj v modelu s SES predstavlja regresijska konstanta (tj. mu oz. mu_j)?
```

O konvergenci:

```
effectiveSize(samples2)
```

```
##            beta   betaGroups[1]   betaGroups[2]   betaGroups[3]   betaGroups[4]
##       2028.5378       3131.6600       1374.2078       3951.5919       8814.9578
##   betaGroups[5]   betaGroups[6]   betaGroups[7]   betaGroups[8]   betaGroups[9]
##       2771.9113       7725.0688       9062.4838       2797.3173      10000.0000
##  betaGroups[10]  betaGroups[11]  betaGroups[12]  betaGroups[13]  betaGroups[14]
##       8361.1982       6718.3056       1432.4114       3882.2729       9167.6880
##  betaGroups[15]  betaGroups[16]  betaGroups[17]  betaGroups[18]  betaGroups[19]
##       3387.0278       6743.5922       4098.6035       2743.6255       9691.4064
##  betaGroups[20]  betaGroups[21]  betaGroups[22]  betaGroups[23]  betaGroups[24]
##       2453.5189       8386.1077       4982.5989       1951.6410       5493.4961
##  betaGroups[25]  betaGroups[26]  betaGroups[27]  betaGroups[28]  betaGroups[29]
##       6178.5100       8323.3641       1746.0607       3004.0402       7782.0669
##  betaGroups[30]  betaGroups[31]  betaGroups[32]  betaGroups[33]  betaGroups[34]
##       8838.1626       4237.6226       9175.8549       8553.3150       2049.3261
##  betaGroups[35]  betaGroups[36]  betaGroups[37]  betaGroups[38]  betaGroups[39]
##       1574.6828       8022.9455       6448.8222       3127.3536       4716.1136
##  betaGroups[40]  betaGroups[41]  betaGroups[42]  betaGroups[43]  betaGroups[44]
##       8962.4191       9576.3382      10000.0000       3336.9100        872.8712
##  betaGroups[45]  betaGroups[46]  betaGroups[47]  betaGroups[48]  betaGroups[49]
##       8364.2294       1965.9632       2242.0488       8487.6932       9129.3705
##  betaGroups[50]  betaGroups[51]  betaGroups[52]  betaGroups[53]  betaGroups[54]
##       8498.1951       7972.1037       1161.2065       5888.9422       2880.5119
##  betaGroups[55]  betaGroups[56]  betaGroups[57]  betaGroups[58]  betaGroups[59]
##       5348.7568       8638.4158       4304.3795       9254.5129       1616.9641
##  betaGroups[60]  betaGroups[61]  betaGroups[62]  betaGroups[63]  betaGroups[64]
##      10000.0000       8275.5022       8966.8316       3347.1253       7779.2697
##  betaGroups[65]  betaGroups[66]  betaGroups[67]  betaGroups[68]  betaGroups[69]
##       9147.3485       8819.7762       9073.5739       8456.7042       3205.6088
##  betaGroups[70]  betaGroups[71]  betaGroups[72]  betaGroups[73]  betaGroups[74]
##       9405.5041       5966.8626       7640.8602       8790.5519       9234.1598
##  betaGroups[75]  betaGroups[76]  betaGroups[77]  betaGroups[78]  betaGroups[79]
##       1605.9499       8797.8501       9333.6366       9252.3820        555.5122
##  betaGroups[80]  betaGroups[81]  betaGroups[82]  betaGroups[83]  betaGroups[84]
##       7694.3377       2716.0563       9268.5041       1790.5826       8072.8323
##  betaGroups[85]  betaGroups[86]  betaGroups[87]  betaGroups[88]  betaGroups[89]
##       2641.9893       9268.4456       5179.5818       8238.7753       8774.7094
##  betaGroups[90]  betaGroups[91]  betaGroups[92]  betaGroups[93]  betaGroups[94]
##       2684.2828       3838.0335       9089.0299       9556.0321       9875.3744
##  betaGroups[95]  betaGroups[96]  betaGroups[97]  betaGroups[98]  betaGroups[99]
```

```
##       5073.0927       8416.5058       5907.4144       4511.6679       9552.1535
## betaGroups[100]             eta         etaBeta              mu     muGroups[1]
##      10000.0000       1418.9850        178.3378       7226.7749      10351.3696
##     muGroups[2]     muGroups[3]     muGroups[4]     muGroups[5]     muGroups[6]
##       9451.0178      10000.0000      10000.0000       8777.1401       9019.2518
##     muGroups[7]     muGroups[8]     muGroups[9]    muGroups[10]    muGroups[11]
##       9520.9568      10000.0000       9674.3348       9654.9065       9527.1464
##    muGroups[12]    muGroups[13]    muGroups[14]    muGroups[15]    muGroups[16]
##      10000.0000      10000.0000       9049.0726       9051.7620      10000.0000
##    muGroups[17]    muGroups[18]    muGroups[19]    muGroups[20]    muGroups[21]
##       8580.3942      10000.0000       9189.0962      10000.0000      10487.3558
##    muGroups[22]    muGroups[23]    muGroups[24]    muGroups[25]    muGroups[26]
##      10000.0000      10000.0000      10000.0000      10000.0000       9697.7864
##    muGroups[27]    muGroups[28]    muGroups[29]    muGroups[30]    muGroups[31]
##      10000.0000      10321.6758       9228.0165       9612.1727      10000.0000
##    muGroups[32]    muGroups[33]    muGroups[34]    muGroups[35]    muGroups[36]
##       9516.1729      10000.0000      10412.9676      10000.0000      10000.0000
##    muGroups[37]    muGroups[38]    muGroups[39]    muGroups[40]    muGroups[41]
##      10305.0232      10000.0000      10000.0000      10760.1171      10000.0000
##    muGroups[42]    muGroups[43]    muGroups[44]    muGroups[45]    muGroups[46]
##       9715.7832       9312.6842      10000.0000      10000.0000      10000.0000
##    muGroups[47]    muGroups[48]    muGroups[49]    muGroups[50]    muGroups[51]
##       9628.8543       9616.1173       9551.3761       9576.5656       8315.4688
##    muGroups[52]    muGroups[53]    muGroups[54]    muGroups[55]    muGroups[56]
##      10000.0000      10000.0000      10000.0000      10000.0000      10000.0000
##    muGroups[57]    muGroups[58]    muGroups[59]    muGroups[60]    muGroups[61]
##      10000.0000      10913.8437       9389.9090      10000.0000       9658.5717
##    muGroups[62]    muGroups[63]    muGroups[64]    muGroups[65]    muGroups[66]
##      10000.0000      10000.0000       9708.0995      10000.0000      10000.0000
##    muGroups[67]    muGroups[68]    muGroups[69]    muGroups[70]    muGroups[71]
##       6873.1717      10109.2415      10000.0000      10000.0000      10000.0000
##    muGroups[72]    muGroups[73]    muGroups[74]    muGroups[75]    muGroups[76]
##       9584.3685      10000.0000       9583.7674      10000.0000      10000.0000
##    muGroups[77]    muGroups[78]    muGroups[79]    muGroups[80]    muGroups[81]
##       9697.1630       9613.3932       7892.8819      10000.0000       9676.3935
##    muGroups[82]    muGroups[83]    muGroups[84]    muGroups[85]    muGroups[86]
##      10000.0000      10000.0000      10000.0000      10000.0000      10000.0000
##    muGroups[87]    muGroups[88]    muGroups[89]    muGroups[90]    muGroups[91]
##      10000.0000      10000.0000       9237.4984      10000.0000      10000.0000
##    muGroups[92]    muGroups[93]    muGroups[94]    muGroups[95]    muGroups[96]
##      10000.0000      10985.6634      10000.0000      10000.0000      10000.0000
##    muGroups[97]    muGroups[98]    muGroups[99]   muGroups[100]           sigma
##      10000.0000      10000.0000      10000.0000      10514.0622       1963.1633
```

```
head(
  sort(
    effectiveSize(samples2))) #najmanjsi izmed effective sample size (etaBeta) - ni id
```

```
##        etaBeta betaGroups[79] betaGroups[44] betaGroups[52]  betaGroups[2]
##       178.3378       555.5122       872.8712      1161.2065      1374.2078
##            eta
##      1418.9850
```

```
#cor(samples2)  neuporabno, ker je izpis predolg
```

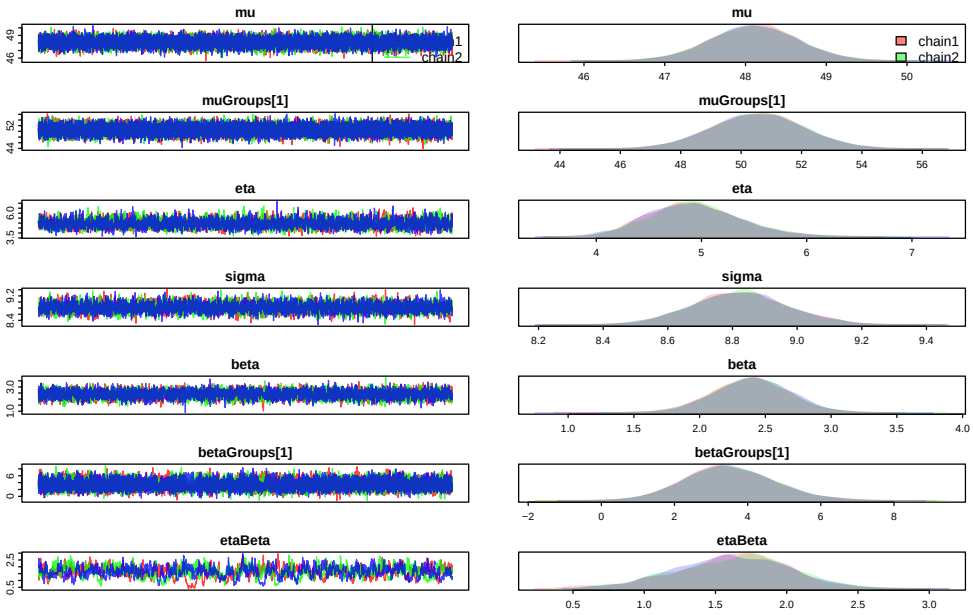
```
head(
  sort(
    abs(
      cor(samples2)[cor(samples2)!=1]), decreasing = T)) #ni idealno, naj bo
```

```
## [1] 0.5488768 0.5488768 0.4791163 0.4791163 0.4177863 0.4177863
```

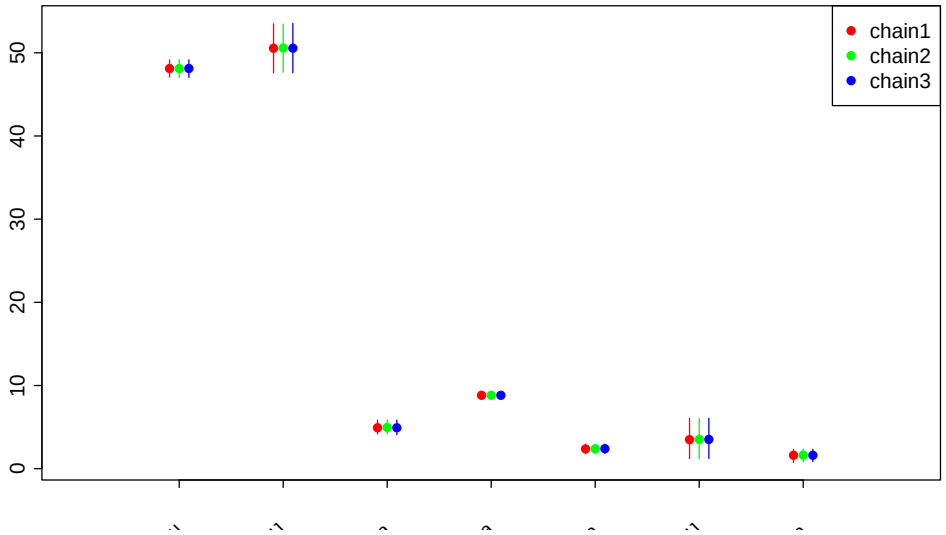
```
# Pogledamo vec verig:
initsFunction2 <- function(){
  list(mu = rnorm(1, mean = mean(pod.sole$povprecje), sd = 10),
       eta = runif(1, min = 0, max = 10),
       sigma = runif(1, min = 0, max = 10),
       muGroups = rnorm(m, mean = pod.sole$povprecje, sd = 10),
       betaGroups = rnorm(m),                  #novo
       beta = rnorm(1),                        #novo
       etaBeta = runif(1, min = 0, max = 10)) #novo
}

samplesList2 <- runMCMC(Cmcmc2, niter = 12000, nburnin = 2000,
                        nchains = 3, inits = initsFunction2)
```

```
## running chain 1...
## |-------------|-------------|-------------|-------------|
## |-------------------------------------------------------|
## running chain 2...
## |-------------|-------------|-------------|-------------|
## |-------------------------------------------------------|
## running chain 3...
## |-------------|-------------|-------------|-------------|
## |-------------------------------------------------------|
```

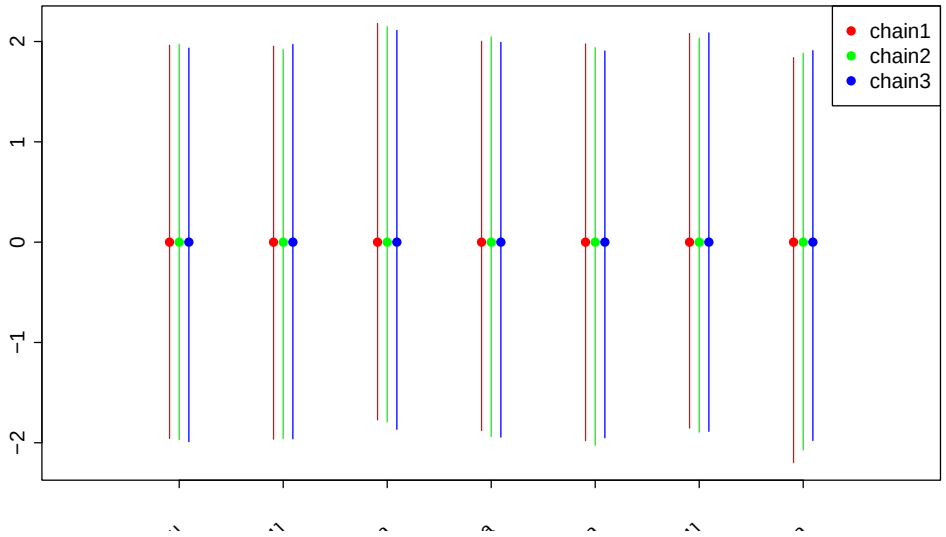
```
chainsPlot(samplesList2, var = c("mu", "muGroups[1]", "eta", "sigma",
                                 "beta", "betaGroups[1]", "etaBeta")) #ok, vendar nekol
```




```
chainsSummary(samplesList2, buffer.left = 1, buffer.right = 1, scale = FALSE,
              var = c("mu", "muGroups[1]", "eta", "sigma",
                      "beta", "betaGroups[1]", "etaBeta")) #ok
```

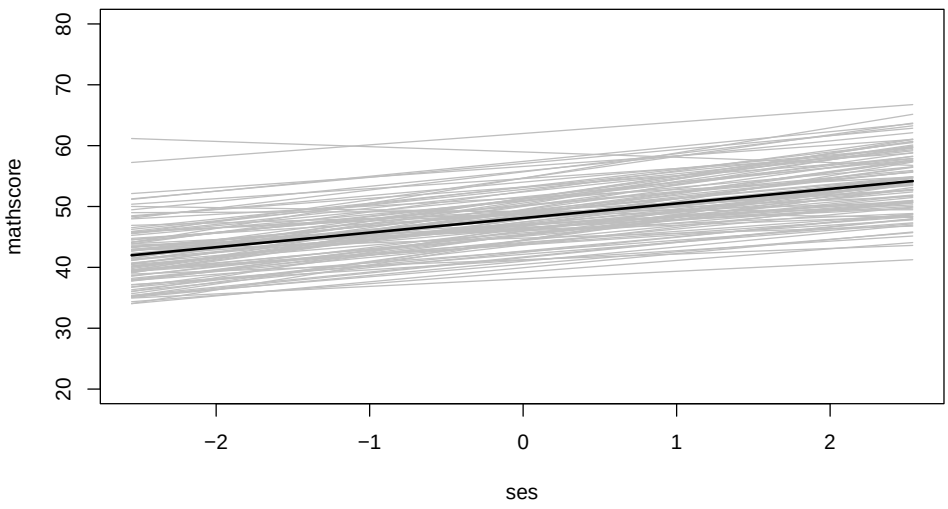



```
chainsSummary(samplesList2, buffer.left = 1, buffer.right = 1, scale = TRUE,
              var = c("mu", "muGroups[1]", "eta", "sigma",
                      "beta", "betaGroups[1]", "etaBeta")) #ok
```



Graficno predstavimo se, kaj smo pravzaprav ocenjevali, tj. - regresijsko premico za vsako solo (mu_j + beta_j * x) in - "regresijsko premico, iz katere so te izhajale" (mu + beta * x).

```
x = c(min(pod2$ses), max(pod2$ses)) #razpon x na podatkih
y = samplesSummary(samples2)[104, 1] + samplesSummary(samples2)[1, 1] * x
#tj. mu + beta * x, kjer mu in beta ocenimo s povprecjem vzorca iz aposteriorne

### Do not run or be patient :)
plot(x, y, type="l", xlab = "ses", ylab = "mathscore", ylim = c(20,80))
for (i in 1:m) {
  y1 <- samplesSummary(samples2)[104+i, 1] + samplesSummary(samples2)[1+i, 1] * x
  #tj. mu_j + beta_j * x, kjer mu_j in beta_j ocenimo s povprecjem vzorca iz aposterio
  lines(x, y1, col = "grey")
}
lines(x, y, col = "black", lwd=2)
```



Bistvo: Model lahko hitro spremenimo, brez teoreticnih izpeljav. Ob tem moramo preuciti konvergenco, po potrebi spremeniti vzorcevalnike.

Zelo zazeljeno (seminarska naloga, 2. naloga): Potrudite se z interpretacijo vecine parametrov (na tem primeru bi si lahko poleg hiperparametrov pogledali se, na primer, v kateri soli ima SES najvecji oz. najmanjsi efekt). Poleg ocene (eno stevilo) je seveda bistven credible interval in statisticna znacilnost.